Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2010

Teófilo José Marques Monteiro

Projecto de um analisador de espectros baseado em SDR

Teófilo José Marques Monteiro

Projecto de um analisador de espectros baseado em SDR

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Nuno Miguel Gonçalves Borges de Carvalho, Professor Associado com Agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor José Manuel Neto Vieira, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho aos meus pais pois sem eles nada disto seria possível.

o júri / the jury

presidente / president

Prof. Doutor José Carlos Esteves Duarte Pedro
Professor Catedrático da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Artur Manuel Oliveira Andrade de Moura
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto (Arguente)

Prof. Doutor Nuno Miguel Gonçalves Borges de Carvalho
Professor Associado com Agregação da Universidade de Aveiro (Orientador)

Prof. Doutor José Manuel Neto Vieira
Professor Auxiliar da Universidade Aveiro (Co-orientador)

agradecimentos / acknowledgements

Gostaria de agradecer aos meus orientadores, Prof. Nuno Borges de Carvalho e Prof. José Vieira, pelo conhecimento e motivação transmitido ao longo deste trabalho. Sem a ajuda e colaboração deles este trabalho não seria realizável.

Também gostaria de agradecer ao senhor Paulo Dias, técnico do Instituto de telecomunicações, pela sua paciência e disponibilidade que sempre demonstrou na montagem e alterações da placa.

Não poderia deixar de agradecer ao meu colega João Carreira, não só pela paciência e amizade, mas pela ajuda que prestou no desenvolvimento deste projecto.

Queria também agradecer em especial a todos os meus colegas desta jornada de 5 anos: Anabela Damas, Bruno Carvalho, Fernando Parente, Jorge Cubal, José Benzinho, Marina Jordão, Isabel Barros, Ana Carlos, Pedro Oliveira, Luis Carlos, Tiago Silva, Gabriel Blard, Filipe Rosário, Carina Moura, Barbara Almeida, Alina Trifan, Miguel Martins, e a todos aqueles que contribuíram para quem sou e me apoiaram ao longo destes anos. Sem eles não teria chegado a este momento da minha vida.

Agradeço ainda à Carolina pela paciência, carinho e grande amizade que me deu ao longo destes 4 anos e a quem devo grande parte da motivação e inspiração para acabar o curso.

Por fim, queria agradecer à minha família, em especial aos meus pais e ao meu irmão por tudo o que me deram e fizeram por mim ao longo da minha vida. Sem o apoio incansável, o carinho e todo o amor que me deram nada disto seria possível.

A todos um grande Bem Haja!

Palavras-chave SDR, Software Defined Radio, Analisador de espectros, USB, ADC

Resumo

Esta dissertação tem como objectivo criar um analisador de espectros baseado em SDR, de baixo custo, utilizando como processador um PC.

O trabalho foi divido em duas fases no entanto apesar do objectivo principal permanecer o mesmo, as bandas a analisar são distintas.

Para realizar a comunicação entre o PC e o front-end foi desenvolvida para cada parte do trabalho uma placa. As placas para além de conterem um módulo USB, incluem ainda ADCs para fazerem ou a digitalização do sinal IF ou de um sinal proveniente de um detector de potência.

Foi ainda desenvolvido, um interface gráfico onde é feito o controlo do front-end e o plot das componentes espectrais do espectro. Este interface permite observar o espectro com algum detalhe, limitado pela resolução imposta pelo hardware.

Key Words SDR, Software Defined Radio, Spectrum Analyzer, USB, ADC

Abstract

This dissertation main purpose is the development of an spectrum analyser low cost, using an PC for processing.

This work is divided in two phases, however the main objective of building an spectrum analyser remains, they analyze diferent bands of spectrum.

To perform the communication between the PC and front-end, an board was developed for each phase. The boards, apart from containing a USB module, also include ADCs to make the digitalization of or the IF signal or the signal from the power detector.

It was also developed an grafical interface where the control of the front-end is made, and the plot of the spectral components of spectrum. This interface also has the ability of show the spectrum with some detail, limited by the resolution imposed by hardware.

Conteúdo

Conteúdo i

Lista de Figuras iii

Lista de Tabelas vii

Acrónimos ix

1 Introdução 1
- 1.1 Enquadramento e Motivação . . . 1
- 1.2 Objectivos . . . 2
- 1.3 Estrutura da dissertação . . . 2

2 Software Defined Radio 3
- 2.1 Introdução . . . 3
- 2.2 O que é um SDR? . . . 3
- 2.3 Arquitecturas de SDR . . . 5
  - 2.3.1 Digitalização em Banda Base . . . 6
  - 2.3.2 Digitalização em Frequências Intermedias (IF) . . . 6
  - 2.3.3 Digitalização em Rádio Frequência (RF) . . . 6
- 2.4 Vantagens do Software Defined Radio (SDR) . . . 7
- 2.5 ADCs . . . 8
- 2.6 Cognitive Radio . . . 9

3 Projecto do analisador de espectros 11
- 3.1 Interface USB 2.0 com o PC . . . 12
  - 3.1.1 USB . . . 12
  - 3.1.2 Esquema geral de comunicação . . . 13
- 3.2 Andar de conversão para IF . . . 14
- 3.3 Conversão A/D utilizando sub-amostragem . . . 15
- 3.4 Análise espectral . . . 18
  - 3.4.1 Estimador de potência . . . 20
  - 3.4.2 Varrimento na frequência . . . 20

4 Implementação do sistema 23
- 4.1 Escolha do controlador USB . . . 23
- 4.2 FT2232H Mini Module . . . 24

4.2.1 Características . . . 25
4.2.2 Chip FT2232H . . . 25
4.2.3 Protocolos . . . 25
4.3 Escolha da ADC . . . 32
4.4 Máquina de estados e lógica adicional . . . 33
4.5 Escolha do downconverter . . . 36
4.5.1 Kit MAX3542EVKIT . . . 36
4.6 Placa de Desenvolvimento para o MAX3542EVKIT . . . 41

5 Interface gráfica e controlo do analisador de espectros 45
5.1 Introdução . . . 45
5.2 Linguagens e Bibliotecas . . . 45
5.2.1 Bibliotecas FTDI . . . 45
5.2.2 Visual C++/CLI . . . 47
5.2.3 NPLOT . . . 47
5.2.4 FFTW . . . 47
5.3 Interface Gráfico e implementação . . . 49
5.4 Controlo do sintetizador . . . 50
5.5 Controlo do módulo MAX3542EVKIT . . . 51
5.6 Controlo da amostragem . . . 53
5.7 Processamento do sinal . . . 53
5.8 Algoritmo de controlo . . . 54

6 Resultados 57
6.1 Placa com a ADC121S101 . . . 57
6.2 Placa com a ADC08100 . . . 59
6.2.1 Maquina de estados . . . 59
6.2.2 ADC . . . 59
6.3 Analisador de espectros . . . 60

7 Conclusões 71
7.1 Trabalho Futuro . . . 71

A Esquemas eléctricos 73
A.1 Placa com a ADC08100 . . . 73
A.1.1 Esquema eléctrico da placa com a ADC08100 . . . 74
A.1.2 Layout da PCB da placa com a ADC08100 . . . 76
A.2 Placa com a ADC121S101 . . . 77
A.2.1 Esquema eléctrico da placa com a ADC121S101 . . . 78
A.2.2 Layout da PCB da placa com a ADC121S101 . . . 78

B Biblioteca NPLOT 79

C Manual de utilização do interface gráfico 83

D Sintetizador 85

Bibliografia 87

Lista de Figuras

2.1 Modelo ideal de um SDR 4
2.2 SDRs da Ettus Research 4
2.3 FLEX-5000C da FlexRadio Systems 5
2.4 SDR-5001 da Spectrum Sinal Processing 5
2.5 Digitalização em Banda Base 6
2.6 Digitalização em IF 7
2.7 Digitalização em banda base 7
2.8 Arquitecturas, aplicações, resolução e frequências de amostragem de ADCs [14] 9
2.9 Evolução temporal da tecnologia de radio, extraído de [19] 10

3.1 Diagrama de blocos do sistema 11
3.2 Esquema de comunicação entre o Personal Computer (PC) e o Front-End . . 14
3.3 Esquema do front-end com mais pormenor 14
3.4 Transformações no mixer 15
3.5 Bloco de amostragem 16
3.6 Subamostragem de IF 17
3.7 Subamostagem 17
3.8 Divisão do espectro em n bandas de observação 18
3.9 Divisão do espectro de observação em k bandas de observação 19
3.10 Divisão real do espectro de observação 19
3.11 Divisão do espectro com as frequências centrais 21

4.1 Módulo UM245R, retirado de [21] 23
4.2 FT2232H Mini Module, retirado de [22] 24
4.3 Diagrama de blocos FT2232H, retirado de [23] 26
4.4 Barramento SPI com um Master e um Slave, retirado de [25] 26
4.5 Barramento SPI com um Master e vários Slaves, retirado de [25] 27
4.6 Diagramas temporais, retirado de [24] 27
4.7 Barramento I^2C com um Master e um Slave[25] 28
4.8 Barramento I^2C com um Master e muitos Slaves[25] 29
4.9 Diagrama temporal de uma transferência, retirado de [26] 29
4.10 Diagrama temporal de leitura e escrita no modo FT245 Style Synchronous FIFO do FT2232H Mini module, retirado de [23] 31
4.11 Diagrama temporal de uma conversão de uma Analog-to-Digital converter (ADC), retirado de [28] 33
4.12 Diagrama de estados 34

4.13 Máquina Estados . . . 35
4.14 Diagrama temporal do circuito proposto . . . 35
4.15 Diagrama funcional do MAX3542, retirado do [29] . . . 37
4.16 Diagrama funcional do kit MAX3542EVKIT, adaptado de [29] . . . 40
4.17 Resposta em frequência do filtro, retirado de [30] . . . 40
4.18 Esquema de ligação KIT MAX3542EVKIT, ADC e Controlador USB . . . 41
4.19 PCB . . . 42
4.20 Placa final . . . 43

5.1 Arquitectura do Windows CDM Driver, retirada de [31] . . . 46
5.2 Exemplo de um gráfico utilizando o NPLOT [32] . . . 48
5.3 Interface gráfico . . . 49
5.4 Esquema de utilização da biblioteca FFTW . . . 50
5.5 Esquema do sintetizador do kit MAX3542EVKIT . . . 51
5.6 Diagrama temporal da transferência de dados da ADC121S101, retirado de [33] 53
5.7 Diagrama de blocos do controlo . . . 54
5.8 Diagrama de blocos do controlo efectuado . . . 55

6.1 Tensão lida no Voltímetro vs ADC . . . 58
6.2 Erro absoluto . . . 58
6.3 Erro relativo . . . 59
6.4 Diagrama temporal da maquina de estados implementada . . . 60
6.5 Sinal sinusoidal de 500kHz com uma potência de -3dBm aplicado a entrada da placa . . . 61
6.6 Sinal sinusoidal de 500kHz com uma potência de -3dBm aplicado a entrada da placa com mais pormenor . . . 61
6.7 Analisador de espectros com um sinal de entrada de 101MHz e -40dBm . . . 62
6.8 Analisador de espectros com um sinal de entrada de 101MHz e -50dBm . . . 62
6.9 Analisador de espectros com um sinal de entrada de 101MHz e -60dBm . . . 63
6.10 Analisador de espectros com um sinal de entrada de 101MHz e -70dBm . . . 63
6.11 Analisador de espectros com um sinal de entrada de 101MHz e -20dBm . . . 64
6.12 Analisador de espectros com um sinal de entrada de 101MHz e -30dBm . . . 65
6.13 Analisador de espectros com um sinal de entrada de 101MHz e -80dBm . . . 65
6.14 Analisador de espectros com um sinal de entrada de 101MHz e -75dBm . . . 66
6.15 Analisador de espectros com um sinal de entrada de 91MHz e -50dBm . . . 66
6.16 Analisador de espectros com um sinal de entrada de 92MHz e -50dBm . . . 67
6.17 Analisador de espectros com um sinal de entrada de 95MHz e -50dBm . . . 67
6.18 Comportamento do sistema para as várias frequências mantendo a potência do sinal constante . . . 68
6.19 Analisador de espectros com um sinal de entrada de 95MHz e -50dBm . . . 69

B.1 Criar um novo projecto . . . 80
B.2 Adicionar a biblioteca NPLOT . . . 80
B.3 Resultado da adição da biblioteca NPLOT . . . 81
B.4 Resultado final . . . 82

C.1 Interface gráfico . . . 83
C.2 Interface gráfico me funcionamento . . . 84

D.1 Diagrama funcional do Si4136 85

Lista de Tabelas

3.1 Os quatro tipo de transferências do USB 13

4.1 Sinais usados no interface FT245 Style Synchronous FIFO 30
4.2 Requisitos temporais do modo FT245 Style Synchrounous FIFO 30
4.3 ADCs série 32
4.4 ADCs 32
4.5 Tabela de estados 34
4.6 Tabela de transições/saídas 35
4.7 Tabela de Excitação 35
4.8 Flip-Flops 35
4.9 Registos de Configuração, retirado do [29] 38
4.10 Tabela ROM, retirado de [29] 38

Acrónimos

2G Segunda Geração

3G Terceira Geração

ADC Analog-to-Digital converter

AGC Automatic Gain Control

API Application Programming Interface

CDM Combined Driver Model

CIL Common Intermediate Language

CLI Common Language Infrastructure

CLR Common Language Runtime

CR Cognitive Radio

DAC Digital-to-analog converter

DFT Discrete Fourier transform

DLL Dynamic-link library

EEPROM Electrically-Erasable Programmable Read-Only Memory

FF Flip-Flops

FFT Fast Fourier transform

FIFO First-in-First-out

FPGA Field-programmable gate array

FTDI Future Technology Devices International Ltd.

GDI Graphical Device Interface

GSM Global System for Mobile Communications

I^2C Inter-Integrated Circuit

IF Frequências Intermedias

IS-95 Interim Standard 95

JTAG Joint Test Action Group

LNA Low Noise Amplifier

LO Local Oscillator

MPSSE Multi-Portocol Synchronous Serial Engine

MSPS Mega Samples Per Second

PC Personal Computer

PCB Printed circuit board

PD Phase Detector

PID Packet ID

PLL Phase-Locked Loop

RF Rádio Frequência

ROM Read Only Memory

SDR Software Defined Radio

SERDES Serialise - Deserialise

SINAD Sinal-to-Noise and Distortion Ratio

SNR Relação Sinal Ruído

SPI Serial Peripheral Interface

UART Universal asynchronous receiver/transmitter

USB Universal Serial Bus

UTMI Universal Transceiver Macrocell Interface

VCO Voltage-Controlled Oscillator

VCP Virtual COM port

Capítulo 1

Introdução

1.1 Enquadramento e Motivação

Com a emergente quantidade de protocolos, standards e modelações surge a necessidade de criar um radio capaz de compreender todos ou quase todos eles. Observando a tabela de distribuição do espectro radioelétrico[1] percebe-se que existe uma quantidade infindável de maneiras de interagir com o meio e que para a mesma banda existem várias maneiras de modelar o sinal para se adequar ao canal.

A complexidade das redes exige esta quantidade enorme de protocolos e interfaces com o meio. Um exemplo desta complexidade são as redes móveis. A Segunda Geração (2G) tinha de suportar vários standards tal como GSM, IS-95, PDC, iDEN. A partir destes standards foram criados novos standards ou serviços de rede aumentando assim a complexidade dos sistemas. Com a introdução da Terceira Geração (3G) os sistemas ficaram ainda mais complexos tornando ainda mais difícil a implementação de um sistema capaz de suportar todos os standards. Se se quiser que todos estes standards sejam implementados serão necessários vários chips para conseguir essa implementação, o que se traduziria numa maior área ocupada pelo sistema e isso não é desejável.

Há portanto, uma necessidade crescente por criar rádios que consigam suportar vários protocolos e se consigam reconfigurar dependendo do serviço e das condições do canal. Chamam-se a estes rádios: SDR ou traduzido a letra, "Rádios definidos por software". Este conceito foi introduzido por Joseph Mitola em 1991 que definiu este tipo de rádio como sendo um rádio que pode definir a sua maneira de interagir com o meio por software permitindo assim que se reconfigure mudando apenas o seu programa. Este tipo de rádios serão uma boa solução no futuro permitindo usar o espectro de maneira mais eficiente, visto que cada vez mais é difícil arranjar lugar para colocar novos standards.

No entanto estes rádios ainda não podem ser totalmente implementados, devido a grandes limitações físicas. Devido aos grandes benefícios que um sistema como este pode trazer às comunicações, existem anualmente um grande número de investigações neste ramo. Com estas têm vindo a aparecer mais sistemas que são versões aproximadas do SDR. É de esperar que num futuro não muito longínquo seja possível ter rádios puramente SDR.

Um analisador de espectros é um dispositivo que analisa a composição espectral de um meio. Com a introdução deste novo conceito de rádio surge a oportunidade de criar analisadores de espectros mais portáteis. São criadas então grandes oportunidades de negócios, uma vez que a construção destes analisadores de espectros irá permitir uma fácil identificação de

problemas de interferências ou ainda identificar fontes de sinal, usando para o efeito antenas direccionais. É pois de grande interesse que sejam criados este tipo de analisadores.

1.2 Objectivos

Esta dissertação tem como objectivo implementar um analisador de espectros digital simples, barato e que seja capaz de interagir com um PC através de Universal Serial Bus (USB). Esta dissertação será divida em duas frentes de trabalho. Na primeira fase desta dissertação, o analisador a ser construído deverá analisar a gama de 2.125GHz até 2.6GHz. Deverá ser desenvolvida para esta fase uma placa que permita o interface com um front-end que será construído no âmbito de uma outra dissertação de mestrado[2]. Este front-end deverá conter um detector de potência. Numa segunda fase será criado um outro analisador que analise a gama de 87MHz a 112MHz. Será utilizado um kit para fazer a conversão de RF para IF. Será necessário ainda desenvolver uma placa que interaja com este kit.

1.3 Estrutura da dissertação

Esta dissertação encontra-se organizada da seguinte maneira:

- **Software Defined Radio** - Neste capítulo será apresentado de uma forma geral o conceito de SDR. São apresentadas as arquitecturas mais comuns e as vantagens de utilizar um SDR. É também descrito neste capítulo o componente mais importante deste sistema: a ADC. Por fim é ainda apresentada a evolução lógica do SDR: o Cognitive Radio.
- **Projecto do analisador de espectros** - É apresentado neste capítulo o projecto geral do sistema. São introduzidos os blocos funcionais e apresentadas algumas das qualidades necessárias para que o sistema funcione correctamente. A partir destas qualidades serão posteriormente escolhidos os componentes constituintes do sistema.
- **Implementação do sistema** - Feita a apresentação geral do sistema, são apresentados neste capítulo as escolhas efectuadas ao longo do projecto e a razão de ser delas.
- **Interface gráfica e controlo do analisador de espectros** - Neste quinto capítulo é apresentada a escolha da linguagem de programação utilizada para realizar o interface gráfico. São apresentadas as bibliotecas utilizadas e como as utilizar. Por fim descreve-se o controlo do sistema.
- **Resultados** - São feitos vários testes neste capítulo permitindo caracterizar o sistema. É verificado o funcionamento do sistema e apresentados os vários resultados.
- **Conclusões** - Neste ultimo capítulo são retiradas as conclusões de todo o trabalho efectuado e é apresentada uma proposta futura de trabalho.

Capítulo 2

Software Defined Radio

2.1 Introdução

Nas ultimas duas décadas os sistemas de comunicações sem fios têm crescido substancialmente. Com o emergir destes, um número infindável de novos *standards* e protocolos apareceu por todo o mundo dificultando o *roaming* através das diferentes regiões geográficas. Com a constante criação de *standards* e protocolos, os sistemas actuais no futuro tornar-se-ão antiquados, tendo surgido assim a necessidade de criar rádios que sejam *future-proof*. Um dos motivos que levou à investigação deste tipo de rádio é o problema de redesenhar o sistema. Este redesenho é custoso, demorado e inconveniente para os utilizadores finais. Os rádios *future-proof* mantêm o *software* e o *hardware* actualizados, à medida que novos interfaces (modulação, frequência, largura de banda do canal), protocolos e tecnologias ficam disponíveis, resolvendo assim o problema de redesenho e garantido que o rádio se mantém actualizado. A partir desta necessidade de criar rádios reconfiguráveis surgiram os *rádios definidos por Software*(**SDR**).

2.2 O que é um SDR?

O termo surgiu pela primeira vez no trabalho realizado por Joseph Mitola em 1991 para caracterizar a classe dos rádios reprogramáveis e reconfiguráveis. Eis uma definição:

> *"A software radio is a radio whose channel modulation waveforms are defined in software. That is, waveforms are generated as sampled digital signals, converted from digital to analog via a wideband DAC and then possibly upconverted from IF to RF. The receiver, similarly, employs a wideband ADC that captures all of the channels of the software radio node. The receiver then extracts, downconverts and demodulates the channel waveform using software on a general purpose processor."[3]*

Joseph Mitola define, como se pode ler, o SDR ideal como um rádio que implementa o seu interface rádio por *software*. Estes rádios são altamente reconfiguráveis e adaptáveis ao meio onde se encontram. O comportamento destes rádios pode variar consoante o *software*. O modelo proposto pelo senhor Mitola é o apresentado na figura 2.1. Pela figura constata-se que o front-end do rádio é apenas constituído por um ADC e por um Digital-to-analog

converter (DAC) que fazem respectivamente a conversão de RF para digital e de digital para RF.

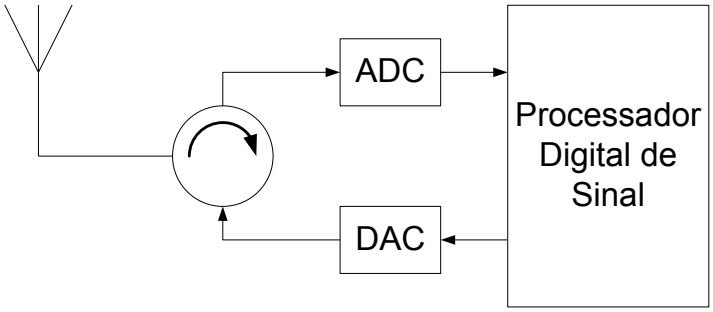


Figura 2.1: Modelo ideal de um SDR

No entanto este SDR ideal não é realizável actualmente, nem tão pouco num futuro próximo, devido ao estado da arte dos componentes (ADCs,DACs,processadores de sinal digital, antenas, entre outros) constituintes do sistema SDR[4, 5].

Apesar destas limitações, um SDR não tem de ser necessariamente ideal. Uma definição exacta e rigorosa de *SDR* não existe pois não há consenso sobre o nível de capacidade de reconfiguração necessária para o qualificar. No entanto, existe algum no que toca à ideia principal: um SDR é um rádio onde parte da camada física é definida por *software*[4, 5].

Um dos primeiros exemplos de uma implementação de SDR foi o *SPEAKeasy*. Foi um projecto desenvolvido pelo departamento de defesa dos Estados Unidos da América iniciado em 1992 que tinha como objectivo criar um rádio que suportasse uma grande variedade de modelações, codificação de dados , encriptação, entre outros parâmetros para uma gama de frequências de 2MHz a 2GHz[4, 6].No entanto, antes do fim do projecto, foi construído um rádio que cobria apenas a gama de 30MHz a 400MHz.

Há ainda outros exemplos bastante conhecidos de rádios SDR para além do já referido, como: o *Joint Tactical Radio System* baseado na tecnologia do *SPEAKeasy*, cujo projecto foi iniciado em 1998 com o objectivo de integrar todos os sistemas de rádio num só [5, 4, 7];e ainda alguns sistemas comerciais *SDR*, dos quais os mais conhecidos são:

- *Universal Software Radio Peripheral (USRP)*[8] e o *USRP2*[9] da *Ettus Research*(figura 2.2). Estes utilizam a plataforma de SDR, o GNU Radio;

(a) USRP (b) USRP2

Figura 2.2: SDRs da Ettus Research

- A família FLEX-5000 da *FlexRadio Systems* [10] (figura 2.3);

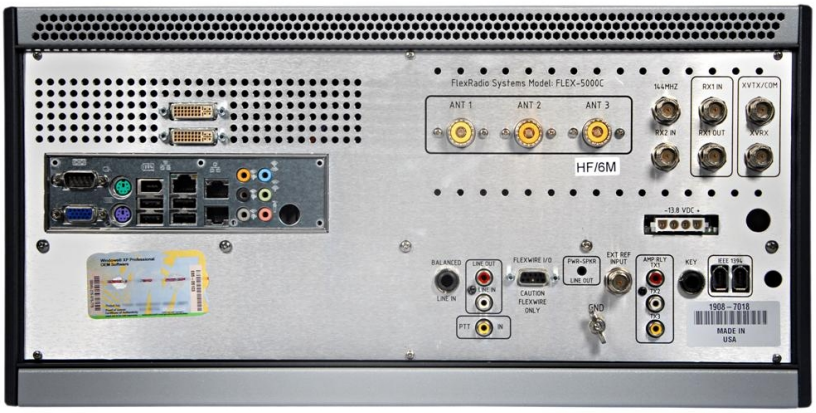


Figura 2.3: FLEX-5000C da FlexRadio Systems

- Da *Spectrum Signal Processing* o rádio *SDR-5001* (figura 2.4)[11];

Figura 2.4: SDR-5001 da Spectrum Sinal Processing

2.3 Arquitecturas de SDR

Um dos pontos chave do SDR é a digitalização. A digitalização consiste numa amostragem de um sinal analógico para uma sequência digital. Esta amostragem é feita à chamada frequência de amostragem, que segundo Nyquist, idealmente não pode ser inferior a 2 vezes a largura de banda. A ADC é o hardware que realiza esta digitalização, tendo como referência uma frequência de amostragem .

Atendendo à posição relativa da digitalização na cadeia de recepção, podem-se definir 3 arquitecturas de SDR: digitalização em Banda Base, digitalização em IF e digitalização em RF.

2.3.1 Digitalização em Banda Base

Nesta primeira arquitectura a digitalização, como o nome indica, é feita em Banda Base e é a mais comum nos dias que correm. Como se pode ver na figura 2.5 o sinal de RF passa para Banda Base através de duas *down conversion*. É na etapa de banda base que é colocada a ADC. Uma das vantagens desta arquitectura é o facto de usar componentes de banda estreita que consomem pouca potência e têm uma grande disponibilidade no mercado. No entanto o uso de componentes de banda estreita limita a capacidade de reconfiguração do rádio pois limita a largura de banda a um canal específico, que por vezes pode não ser suficiente para receber um sinal. Com este conjunto de restrições, o rádio torna-se limitado a uma função específica.

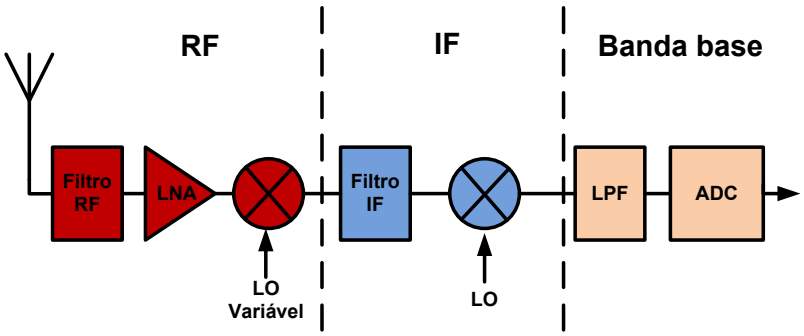


Figura 2.5: Digitalização em Banda Base

2.3.2 Digitalização em IF

Nesta segunda arquitectura, o sinal digitalizado encontra-se em IF. Esta arquitectura é uma aproximação de SDR, no entanto , a digitalização em vez de ser feita em RF é feita em IF. Isto implica que seja adicionado à arquitectura ideal um andar para fazer a down conversion para IF,como se pode ver na figura 2.6. A escolha de IF depende não só da ADC e da sua frequência de amostragem, mas também da capacidade do processador de processar as amostras provenientes da ADC. A diferença desta arquitectura para a arquitectura de digitalização em banda base é a parte da banda base, visto que nesta arquitectura é implementada por *software*, eliminando assim o uso de alguns componentes discretos. Uma das vantagens desta arquitectura é o seu desenho poder ser mais compacto, no entanto tem a desvantagem de aumentar o consumo de energia dos componentes.

2.3.3 Digitalização em RF

A digitalização em RF, é uma aproximação ao SDR ideal. Como se pode ver na figura 2.7 esta arquitectura é muito semelhante ao receptor proposto por Joseph Mitola. Esta arquitectura fornece, como já foi visto antes, grandes desafios, não só porque a ADC deve cumprir uma vasta gama de especificações, o amplificador de baixo ruído deve ser capaz de cobrir todo o espectro e o processador digital deve ter uma enorme capacidade de processamento para acompanhar o número elevado de amostras produzidas pela ADC. No entanto, apesar desta restrições, este rádio deverá ser capaz de cobrir todo o espectro e suportar qualquer tipo de técnica de modelação.

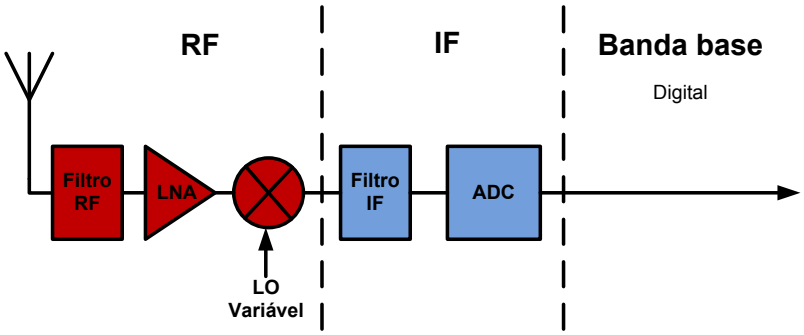


Figura 2.6: Digitalização em IF

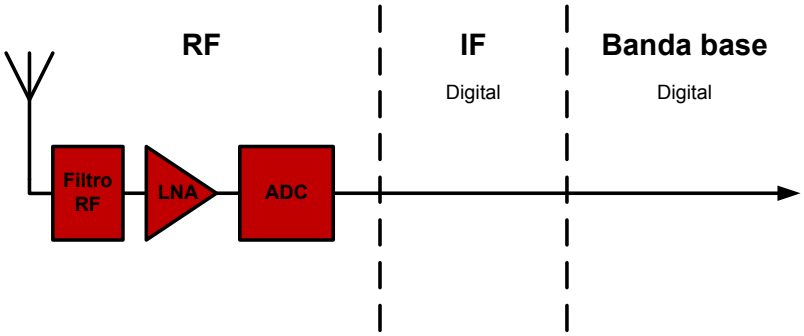


Figura 2.7: Digitalização em banda base

2.4 Vantagens do SDR

Apesar da implementação de um *SDR* ideal ter várias restrições e não ser possível implementá-lo num futuro imediato, a investigação no ramo não pára, isto porque são grandes os benefícios que esta tecnologia pode trazer no futuro.

O SDR tem uma arquitectura bastante flexível que permite mudar facilmente de interface de rádio, em tempo real, através de um simples *download* de *software*. É através de um simples *download* que é possível, também, actualizar o rádio para que este consiga suportar um novo protocolo ou standard. A flexibilidade de um SDR permite ao rádio uma fácil adaptação a ambientes com múltiplos protocolos e standards pois não fica restrito a um só protocolo. A possibilidade de reconfiguração, dá a este tipo de rádio uma vantagem em relação aos rádios actuais, pois poderá ser possível implementar protocolos que não existam em qualquer banda de frequências[12].

Em [13] são apontados 5 factores que vão tornar os SDR mais conhecidos:

- **Multifuncionalidade** - Com o desenvolvimento das tecnologias de comunicação sem fios alguns serviços complementares dos sistemas são distribuídos. No entanto, estes equipamentos estão limitados a um protocolo. A implementação de um SDR num PC permite que este com um simples *download* de *software* suporte o protocolo para interagir com o equipamento de serviços complementares.

- **Mobilidade global** - É bastante fácil de entender que com a quantidade enorme de standards de comunicação sem fios, é difícil um rádio com uma arquitectura convencional

conseguir cobri-los todos. No entanto os *SDRs* são capazes de se reconfigurarem e esta capacidade dá-lhes a vantagem de se poderem adaptar ao ambiente, ou seja, são capazes de interagir com diferentes tipos de protocolos existentes nas diferentes diferentes regiões geográficas.

- **Capacidade de compactação e eficiência energética** - Devido a sua arquitectura simples, o *SDR* consegue ter uma implementação compacta, em alguns casos com eficiência energética, em comparação com as arquitecturas convencionais.

- **Facilidade de manufacturação** - Como se sabe os componentes RF são muito difíceis de estandardizar devido às suas características de desempenho poderem variar muito. Com a digitalização prematura do sinal a quantidade de componentes RF utilizados é reduzida, eliminando assim os problemas introduzidos por estes. Com o número inferior de componentes a manufacturação destes rádios será bastante mais simples que os convencionais.

- **Facilidade de actualizações** - Com o progresso na área das comunicações sem fios os serviços poderão necessitar de actualizações, de modo a que sejam feitas melhorias ao sistema, assim como novos serviços que podem ter que ser incluídos no rádio de modo a mantê-lo actual. Estas actualizações terão de ser feitas em tempo real sem que estas interfiram com o funcionamento da infra-estrutura do rádio. Actualizar um rádio nos dias que correm significa uma mudança de hardware, no entanto com um simples *download* um SDR mantém-se actualizado.

2.5 ADCs

O componente que tem um dos papeis mais importantes na cadeia de recepção do SDR é a ADC.

A constante evolução destes componentes tem permitido que seja possível fazer amostragens de um sinal com um IF mais elevado, sem que se tenha a necessidade de se fazer a passagem para banda base, obtendo-se assim todas as vantagens do domínio digital.

Na figura 2.8 são apresentadas as várias arquitecturas existentes das ADCs. Nesta figura também se podem observar as aplicações, a resolução e as frequências de amostragem atingidas por cada arquitectura de ADC. Hoje em dia existem ADCs que atingem velocidades acima dos GHz mas são demasiado caras e têm um elevado consumo de potência, não constituindo uma boa solução.

Existem dois standards que caracterizam as ADCs que são o IEEE Std 1241-2000[15] e o IEEE Std 1057-1994[16]. Estes dois standards referem as características mais importantes que devem ser alvo de atenção nas ADCs.

Uma das características mais importantes das ADCs é o Relação Sinal Ruído (SNR). Esta característica mede a relação de potência entre o sinal S e o ruído N, isto é, quantas vezes a potência do sinal é maior que o ruído que normalmente é expressa em dB. Na equação 2.1 está expresso matematicamente como é calculado o SNR.

$$SNR = \frac{S}{N} \tag{2.1}$$

Outra maneira de caracterizar uma ADC é o Sinal-to-Noise and Distortion Ratio (SINAD). O SINAD é nada mais que a relação entre a potência do sinal e a soma das potências do

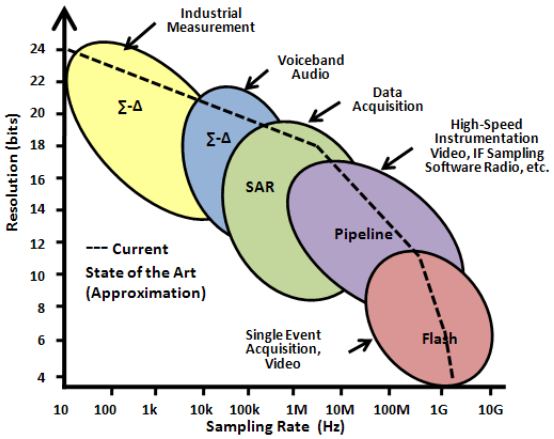


Figura 2.8: Arquitecturas, aplicações, resolução e frequências de amostragem de ADCs [14]

ruído e da distorção, considerando que o sinal é composto pelo sinal com ruído e distorção. Matematicamente isto traduz-se na equação 2.2

$$SINAD = 20log\left(\frac{S}{D+R}\right) \tag{2.2}$$

Estas duas maneiras de caracterizar as ADC ajudam a compreender o que pode acontecer quando um sinal é digitalizado e que tipo de erro lhe vem associado. Quanto maior forem as duas características melhor, pois significa que o sinal está mais imune ao ruído de digitalização e também às distorções.

Estas distorções podem ser provocadas não só por erros nas escalas, isto é, erros de offset e ganho, mas também por distorções das harmónicas e distorções de intermodulação geradas pelo sinal à entrada .

Apesar destas características serem muito importantes o consumo de potência e as velocidades são os factores mais preponderantes. O consumo de potência pode ser até um factor de escolha decisivo visto que se uma ADC tiver um maior consumo de potência que uma ADC mais lenta, em sistemas móveis pode levar a ser escolhida a ADC que menos consuma. Por causa destes factores há muita investigação nos dois ramos: uns investigam sobre como aumentar a frequência de amostragem, outros como diminuir o consumo[5, 17].

2.6 Cognitive Radio

Com a possibilidade de configuração dos rádios uma nova ideia surgiu, os Cognitive Radios (CRs). Foi inventado também por Mitola[18] e é essencialmente um SDR que analisa o espectro em seu redor, sente variações e reage conforme as suas descobertas[18]. Para além disto o CR procura ainda buracos no espectro para transmitir, adaptando a sua modulação consoante à largura de banda e condições do canal. Deverá ser ainda capaz de transmitir sinais de potência que podem ser controlados para reduzir as interferências. Para que tudo isto seja possível, é necessário que o CR tenha um *front end* capaz de cobrir uma enorme quantidade de espectro.

Este tipo de rádio que se adapta ao meio será muito comum num futuro não muito distante. Na figura 2.9 pode-se observar a evolução temporal dos rádios, e quais são as tendências para

o futuro. Como é de esperar a tendência natural dos rádios é o CR.

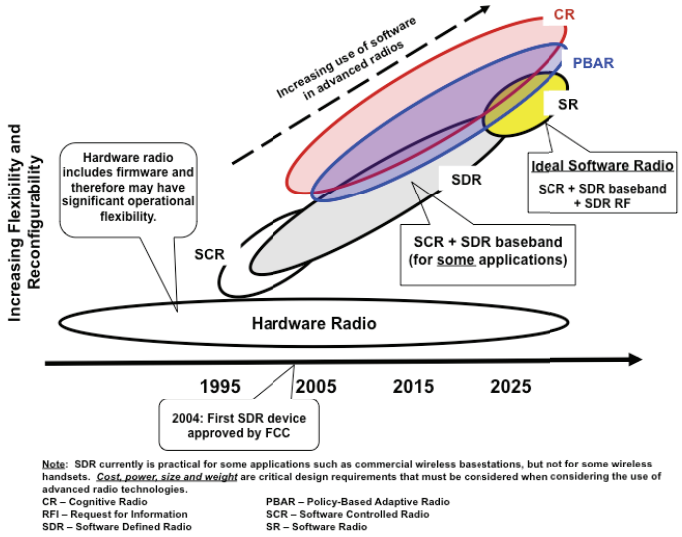


Figura 2.9: Evolução temporal da tecnologia de radio, extraído de [19]

Capítulo 3

Projecto do analisador de espectros

O analisador de espectros a desenvolver nesta dissertação é um sistema baseado em SDR. Pretende-se que esse sistema analise uma pequena gama de frequências. É ainda pretendido que a digitalização seja feita em IF e assim sendo, será necessário que o *front-end* tenha uma arquitectura semelhante à apresentada na subsecção 2.3.2. Nesta dissertação optou-se também por que o processador de sinal que irá fazer o tratamento das amostras provenientes da ADC e o controlo do *front-end* seja um PC.

O sistema a desenvolver foi dividido em blocos funcionais e é apresentado na figura 3.1. Nas próximas secções serão abordados os blocos funcionais.

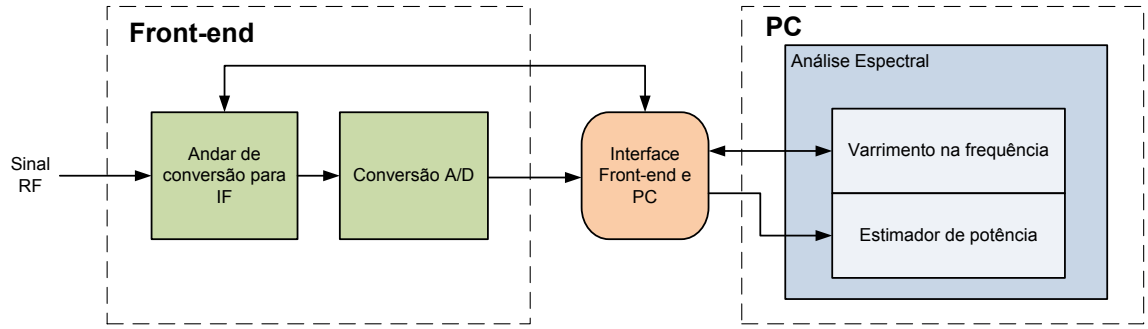


Figura 3.1: Diagrama de blocos do sistema

3.1 Interface USB 2.0 com o PC

3.1.1 USB

O USB é um barramento constituído por um *Host* e um ou vários *devices*. É o *Host* que está encarregado de gerir e interagir com os *devices*. Quando um *device* é ligado fisicamente ao *Host*, este deve detectar esta ligação e interrogar o *device* para determinar quais os drivers apropriados para interagir com o *device*. Este processo é designado por enumeração.

O USB atinge 4 velocidades: *Low speed* (1.5Mbps), *Full speed* (12Mbps), *High speed* (480Mbps) e *Superspeed* (5Gbps).

A comunicação no barramento USB é sempre feita entre o *Host* e um *device*. É o *host* que inicializa sempre uma transferência e o *device* deve responder. É o *host* que gere o tráfico de dados no barramento. As transferências são feitas dos *endpoints* dos *devices*. Cada *endpoint* tem um número, um sentido e um número máximo de bytes de dados associado. Estes são a fonte e/ou destino de dados.

Uma transferência no USB 2.0 começa quando o *host* envia um pacote *token*. Este contém a informação sobre o número de *endpoint* alvo e a direcção da transferência. Um pacote *token* IN pede um pacote de dados ao *endpoint*, enquanto que um pacote *token* OUT precede um pacote de dados vindo do *host* para o *endpoint*. Cada pacote de dados contém ainda um corrector de erros e um Packet ID (PID). Muitas das transferências têm um pacote de *hanshake* para notificar que foi recebido com sucesso o pacote de dados. Cada pacote começa com o PID que contém a informação sobre a transação efectuada.

Para haver transferências de dados é necessário que o *host* e o *device* estabeleçam um *pipe*. Um *pipe* é uma associação entre um *endpoint* de um *device* e um *software* de controlo do *host*. Estes são estabelecidos durante a enumeração, e se algum *device* for removido o *host* elimina o *pipe* que já não vai ser necessário. O *host* tem ainda a possibilidade de poder adicionar ou remover *pipes* através de transferências de *control*. Todos os *devices* têm de ter um *pipe default* de controlo que utiliza o *endpoint* zero.

O USB suporta quatro tipo de transferências: *control*, *bulk*, *interrupt* e *isochronous*. No entanto apenas os três últimos transportam dados.

As transferências do tipo *control* são suportadas por todos os *devices* através do *default pipe* no *endpoint* zero e servem para transportar informação que permite ao *host* obter a configuração do *device*. Este tipo de transferências pode também transportar pedidos definidos por uma classe ou vendedor e é utilizado na enumeração para a melhor escolha do driver que vai ser utilizado pelo *host*.

Este tipo de transferências tem até 3 etapas: *Setup*, *Data* e *Status*. A etapa *Setup* contém um pedido, enquanto que na etapa *Data* são enviados dados do *host* ou do *device* dependendo do tipo de pedido. Na etapa *Status* vem a informação sobre o sucesso da transferência. Pode ou não haver etapa *Data* dependendo do tipo de pedido efeito pelo *host*.

A transferência do tipo *bulk* é útil para transferir dados quando o tempo não é crítico e se querem transferir grandes quantidades de dados sem que seja ocupado o barramento USB durante muito tempo pois este tipo de transferência utiliza o tempo em que o barramento não é utilizado pelos outros tipos de transferências. Quando apenas no barramento é usado este tipo de transferência *bulk* este torna-se o tipo mais rápido de transferência de dados.

As transferências *interrupt* são usadas quando os dados não podem ter *delay*. Não é obrigatário um *device* suportar este tipo de transferências, no entanto é reservada uma largura de banda para que estes dados consigam cumprir os requisitos temporais. Uma transferência

interrupt acaba quando todos os dados foram transferidos.

As transferências *isochronous* são usadas quando os dados têm que ter um ritmo constante de recepção ou dentro de um tempo específico em que os erros são toleráveis. Tal como nas transferências *interrupt* é reservada uma largura de banda que é partilhada com este tipo de transferências. A diferença para os outros tipos de transferências é que não há corrector de erros, pois é necessário garantir que os dados sejam enviados dentro das restrições.

Na tabela 3.1 podem-se ver os tipos de transferências e com mais detalhe algumas das características mais importantes de cada transferência.

Tipo de transferência	**Control**	**Bulk**	**Interrupt**	**Isochronous**
Uso típico	Identificação e configuração	Impressoras, scanners	Ratos e Teclados	Streaming áudio,video
Necessário suporte?	Sim	Não	Não	Não
Permitido em low speed?	Sim	Não	Sim	Não
Tamanho máximo de pacotes (SuperSpeed)	512	1024	1024	1024
Tamanho máximo de pacotes (High Speed)	64	512	1024	1024
Tamanho máximo de pacotes (Full Speed)	64	64	64	1023
Tamanho máximo de pacotes (Low Speed)	8	Não é premitido	8	Não é premitido
Direcção dos dados	IN ou OUT	IN ou OUT	IN ou OUT	IN ou OUT
Largura de banda reservada?	10% em low/full speed; 20% a high speed/ SuperSpeed	Nenhuma	90% a low/full speed, 80% a high speed e SuperSpeed (isochronous e interrupt combinados, no maximo)	
Corrector de erros?	Sim	Sim	Sim	Não

Tabela 3.1: Os quatro tipo de transferências do USB

3.1.2 Esquema geral de comunicação

O que se propõe nesta dissertação é criar um analisador de espectros que seja fácil de usar e acessível para qualquer utilizador, constituindo por isso a utilização do protocolo USB uma mais valia. Na figura 3.2 pode-se ver o esquema genérico proposto.

É através do barramento USB que o PC pode fazer o controlo do front-end e obter o sinal digitalizado IF. É, portanto, importante que não haja falhas na comunicação entre o PC e o USB, pois isso pode levar a que alguns dados sejam mal interpretados ou mesmo perdidos, conduzindo a erros de medição do espectro ou mesmo erros no controlo. O protocolo USB

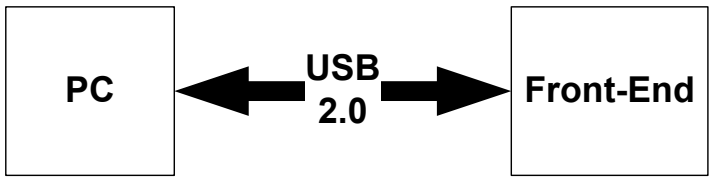


Figura 3.2: Esquema de comunicação entre o PC e o Front-End

tem embutido um corrector de erros, minimizando desta forma os possíveis erros que podem resultar da ligação física.

O front-end deve então conter um chip que faça a comunicação através do USB. O meio mais fácil de se fazer esta comunicação é utilizando módulos USB que já tenham drivers próprios, pois o desenvolvimento de drivers é muito demorado. Há empresas que passam anos a tentar desenvolver um driver de USB e, mesmo assim, em muitos casos eles não funcionam correctamente.

Como já foi dito, este módulo USB deve ser capaz de transmitir as amostras provenientes do bloco de amostragem para o PC e deverá ainda controlar o bloco downconverter.

3.2 Andar de conversão para IF

No andar de conversão para IF é onde o sinal RF é convertido para IF. Na figura 3.3, pode-se observar a constituição do front-end incluindo a secção do downconverter (que é nada mais que um super-heterodino adaptado).

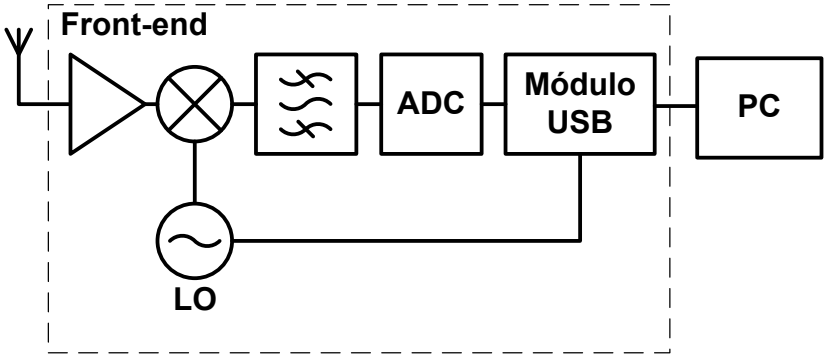
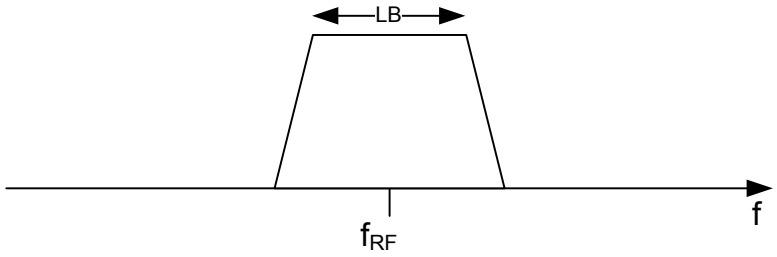


Figura 3.3: Esquema do front-end com mais pormenor

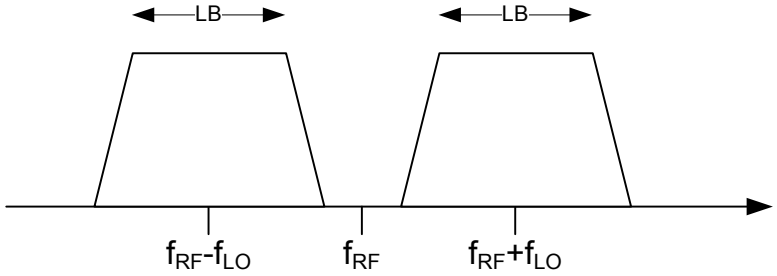
O downconverter, como está representado na figura, é constituído normalmente por um amplificador de preferência de baixo ruído, um mixer, um Local Oscillator (LO) e um filtro passa banda. O oscilador local terá de ser programado e controlado através do USB pelo PC, pois poderá tratar-se de um sintetizador, o qual terá de ser programado para se obter a frequência desejada.

Muitos destes componentes têm limitações de largura de banda, o que leva a que apenas se consiga transformar uma pequena banda de RF para IF. O amplificador de baixo ruído e o mixer são ambos componentes limitadores devido ao facto de serem componentes não lineares. De modo a que não introduzam não linearidades no sistema é muito comum limitar a largura de banda do sinal à entrada destes componentes, evitando assim que estes saturem

ou operem a frequências fora da gama nas quais poderiam provocar distorções no sinal.



(a) Espectro de um sinal RF antes do mixer



(b) Espectro de um sinal RF depois do mixer

Figura 3.4: Transformações no mixer

O elemento que se encarrega de transformar o sinal RF em IF é o mixer. De uma maneira simples, o mixer faz a mistura do sinal RF com o sinal proveniente do oscilador local, que não é mais que um sinal sinusoidal com uma frequência f_{LO}. Esta mistura é nada mais que uma multiplicação. Supondo um sinal à entrada do mixer semelhante ao apresentado na figura 3.4a, isto é, com uma largura de banda de LB e centrado em f_{RF}, da mistura com o sinal proveniente de LO, resulta um sinal que tem um espectro semelhante ao da figura 3.4b. Como se pode perceber pela figura deve-se escolher um f_{LO} de modo a que $f_{RF} - f_{LO}$ ou $f_{RF} + f_{LO}$ centre o espectro do sinal em IF.

O filtro depois do mixer serve para eliminar uma destas imagens. Normalmente, este filtra a imagem do sinal que tem o IF mais baixo. O filtro não elimina só a imagem não desejada introduzida pelo mixer, limita também a largura de banda à entrada do ADC, e portanto a largura de banda do sinal a digitalizar.

3.3 Conversão A/D utilizando sub-amostragem

É neste bloco que é feita a conversão entre o domínio analógico e o domínio digital. Esta conversão é realizada através de um ADC que deverá ter uma frequência de amostragem superior a duas vezes IF. Um outro requisito importante é a largura de banda da ADC. Se esta conseguir abranger IF será uma mais valia pois poderá ser necessário fazer a subamostragem do sinal IF, visto que a velocidade de transmissão de dados do protocolo USB pode delimitar a frequência de amostragem do sinal.

A figura 3.5 é a representação genérica do bloco. Pode-se observar que este bloco necessita de uma frequência de amostragem, representada por f_a, para o funcionamento correcto do

bloco. É a partir desta frequência que o sinal IF é amostrado e quantificado em N bits.

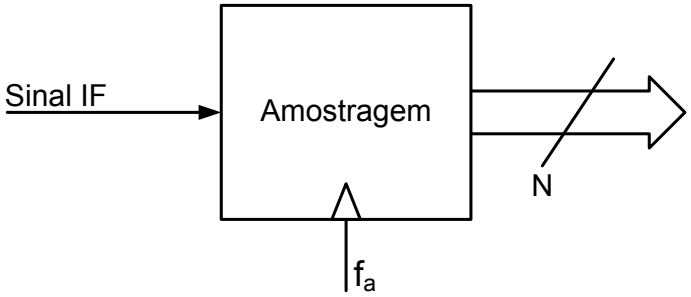


Figura 3.5: Bloco de amostragem

Como já foi referido na secção anterior, a conversão de RF para IF apenas é possível em pequenas bandas devido a limitações de alguns dos componentes constituintes do downcoverter. Devido a este conjunto de limitações, a porção de espectro a analisar, apesar de pequena, não poderá ser digitalizada de uma só vez.

Surge ainda uma outra limitação relacionada com a velocidade do USB, que impõe limitações ao nível da frequência de amostragem do ADC. Estas irão influenciar a largura de banda que o ADC consegue digitalizar. Esta limitação tem uma relação directa com a capacidade de os dados serem transmitidos do módulo para o PC, isto é, se o módulo apenas conseguir enviar dados a uma determinada frequência, não convém que o ADC supere esta frequência de envio pois isso resultaria numa perda de amostras. No entanto, é possível que o ADC trabalhe a uma frequência superior à da velocidade de transmissão de dados. Desde que o módulo tenha buffers para armazenar as amostras provenientes do ADC, este poderá ter uma frequência de amostragem superior. O módulo deverá apenas ser capaz de comunicar com o ADC a uma velocidade superior à velocidade de transmissão de dados para o PC. No entanto, como os buffers têm uma capacidade de armazenamento finita, a amostragem não poderá ser contínua.

Apesar de poder ser possível aumentar a frequência de amostragem acima da velocidade de transmissão, essa poderá não ser suficientemente elevada para cobrir IF. Se não for possível uma f_a para a ADC que seja suficientemente grande para cobrir IF, será necessário fazer uma subamostragem desta banda. Esta subamostragem apenas é possível se a largura de banda do ADC o permitir e se o sinal IF estiver dentro de uma banda de subamostragem, isto é, o sinal IF tem de ter uma largura de banda LB e uma frequência central f_{IF} que obedeçam à seguinte equação:

$$f_a \cdot i < f_{IF} - \frac{LB}{2} < f_{IF} + \frac{LB}{2} < \frac{f_a}{2} \cdot (i+1) \ \wedge \ i \in \mathbb{N}^+ \tag{3.1}$$

Na figura 3.6 é dado um exemplo de um sinal IF que fica centrado numa banda de subamostragem.

A subamostragem é uma boa solução, mas ainda assim acarreta alguns problemas. Um deles é a inversão do espectro de IF, como está representado na figura 3.7. Na subfigura 3.7a estão representadas as várias bandas do sinal antes de ser digitalizado, isto é, à entrada da ADC, e na subfigura 3.7b estão representadas as bandas do sinal depois de digitalizado. É de referir que apenas as bandas pares invertem, isto é, B, D, F, G, etc.. Neste caso apenas B é invertida, passando de B para b, por se tratar de uma banda par. Apesar de tudo, este

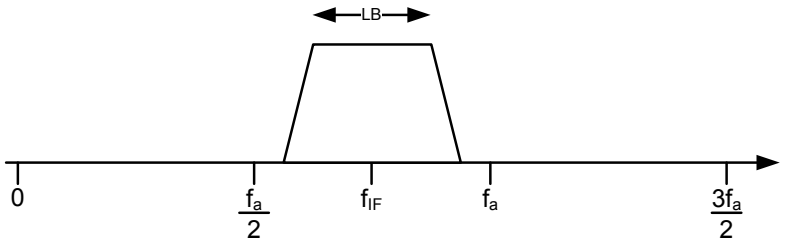
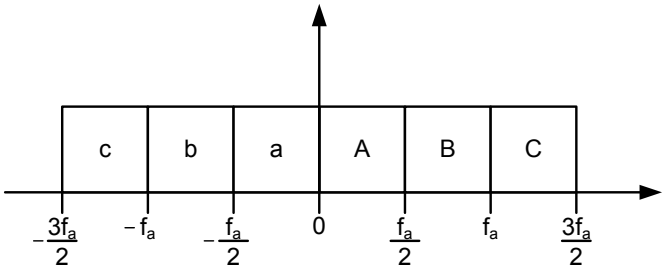
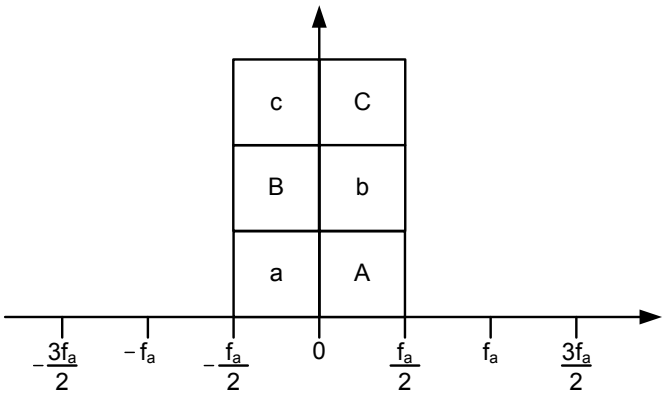


Figura 3.6: Subamostragem de IF



(a)



(b)

Figura 3.7: Subamostagem

problema é facilmente resolvido digitalmente invertendo o espectro, retomando assim a sua forma inicial.

Um outro problema da subamostragem é a sobreposição de bandas, isto é, as várias bandas do sinal caem todas entre 0 e $\frac{f_a}{2}$ o que provoca somas de espectros. O problema em si pode ser minimizado desde que o sinal IF seja filtrado de modo a que apenas a banda IF chegue ao ADC. Deste modo, como o sinal IF tem um espectro mais forte que o resto do espectro, quando digitalizado, o sinal vai ser maioritariamente o sinal IF. No entanto, este filtro tem de ser bastante selectivo e não deixar passar mais do que a banda IF. Se tal não acontecer,o sinal IF, quando digitalizado, poderá conter componentes espectrais de aliasing.

Devido a estes factores, o espectro a analisar terá de ser dividido. Está divisão é feita em

pequenas bandas, tendo estas uma largura igual ou inferior à largura de banda máxima do filtro à entrada do ADC. A esta banda dá-se o nome de banda de observação.

3.4 Análise espectral

Neste bloco é feito o tratamento das amostras provenientes do bloco onde é feita a conversão do domínio analógico para o digital. Devido às limitações referidas na secção anterior, o espectro, vai ter de ser dividido em bandas de observação.

Supõe-se que uma banda de observação de espectro, tem uma largura de banda máxima ΔB, que é igual à imposta pelas limitações dos componentes constituintes do downconverter, ΔB_{max}. Esta largura de banda máxima é n vezes menor do que o espectro a observar, B, ou seja:

$$B = n \cdot \Delta B_{max}, \quad n \in \mathbb{N}^+ \tag{3.2}$$

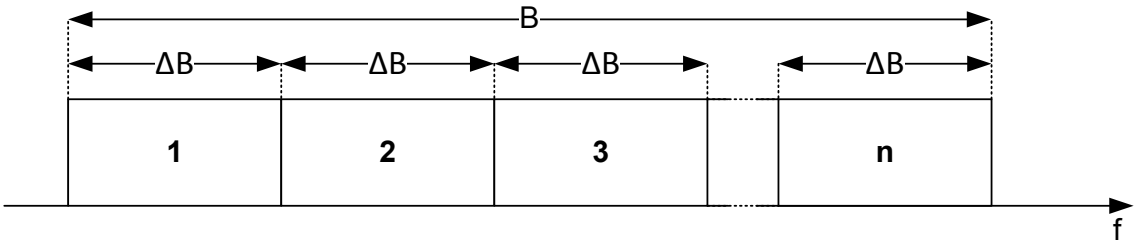


Figura 3.8: Divisão do espectro em n bandas de observação

Da equação 3.2 conclui-se que n é o número de bandas não coincidentes necessárias para cobrir o espectro a observar, isto é, dentro do espectro a observar cabem n bandas de largura ΔB como se pode ver na figura 3.8. Este n corresponde também ao número de saltos em frequência que o oscilador local tem de realizar e, por cada salto, este desloca-se na frequência ΔB. No entanto, no caso de o oscilador não conseguir realizar saltos de ΔB o número de bandas não é um número inteiro. Para garantir que o número de bandas de observação é um número inteiro, deve cobrir-se mais banda para além daquela que é necessário observar. O ΔB escolhido terá de ser menor do que o imposto pelas limitações, ou seja $\Delta B < \Delta B_{max}$, e de preferência o maior salto ΔB realizável pelo LO que obedece à condição anterior. O k que minimiza o número de bandas de observação obedece à seguinte equação:

$$B \leq k \cdot \Delta B, \quad k \in \mathbb{N} \ \wedge \ n \leq k < n+1 \ \wedge \ n \in \mathbb{R}^+ \tag{3.3}$$

A expressão 3.3 também é válida para o caso em que o oscilador consegue fazer saltos ΔB_{max} mas origina um número de bandas de observação que não é inteiro. Também neste caso se cobre mais banda para além daquela que se quer observar.

Tem-se então um espectro que será dividido em k bandas de observação, como se pode ver na figura 3.9. No início da banda de espectros a observar, $f_{B_{ini}}$, é colocado o início da primeira banda de observação garantindo assim que toda a gama de espectro B esteja dentro das bandas de observação.

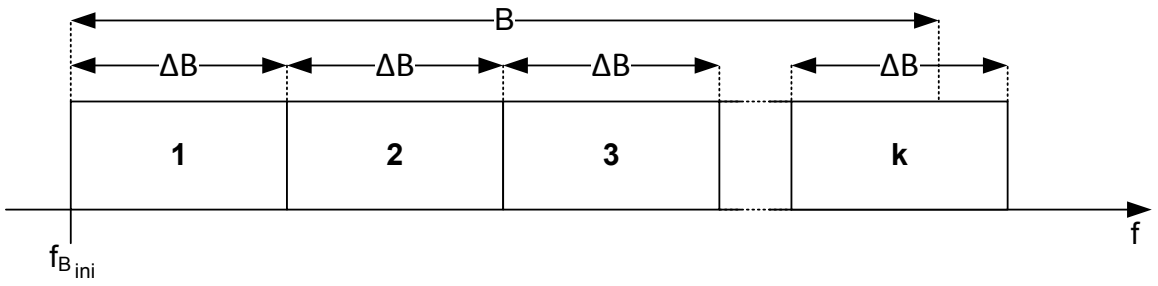


Figura 3.9: Divisão do espectro de observação em k bandas de observação

Os casos apresentados são em tudo diferentes da prática, pois é de notar que as bandas de observação da figura 3.9 são ideais. Isto implicaria que o filtro aplicado à entrada do ADC e depois do mixer tivesse uma resposta em frequência ideal. No entanto, não é possível implementar este tipo de filtro, devido à sua resposta em frequência ideal, tornando-o assim impossível de realizar.

O filtro a colocar à entrada do ADC terá então de ter uma banda de transição menor que a máxima largura de banda que a ADC consegue digitalizar, para evitar *aliasing* na digitalização. Deverá ter-se, portanto, um ΔB o maior possível mas de modo a evitar *aliasing*.

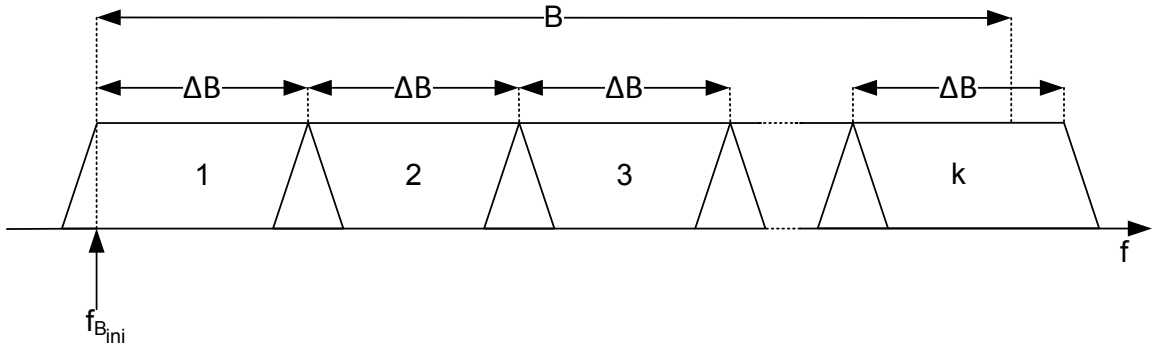


Figura 3.10: Divisão real do espectro de observação

Na figura 3.10 pode-se ver o aspecto real das bandas de observação. Apesar de haver sobreposição nas janelas, é possível ficar-se apenas com a banda de transição da janela, mas isto apenas pode ser realizado em sinais digitalizados. O número de bandas de observação continua a ser calculável a partir das fórmulas 3.3 ou 3.2, dependendo dos casos, mas neste caso é usado um ΔB igual a banda de transição do filtro à entrada da ADC. É de notar que neste caso o início da banda,$f_{B_{ini}}$ a observar é colocado no início da banda de transição.

É a partir da transformada discreta de Fourier que são obtidas as componentes espectrais do sinal, como se pode ver na equação 3.4.

$$f_j = \sum_{k=0}^{n-1} x_k e^{-\frac{2\pi i}{n} jk} \tag{3.4}$$

Cada elemento desta transformada corresponde a uma frequência, ou seja, cada f_j é a componente espectral de uma frequência que pode ser determinada em função da frequência

de amostragem e do número de amostras, $f = \frac{f_a}{N_a} \times N$. Partindo desta propriedade da transformada de Fourier, podemos retirar apenas os dados relativos à componente espectral da banda de transição.

É de notar que para a primeira parte deste trabalho, isto é onde o modulo de front-end será desenvolvido no âmbito de outra dissertação, esta secção é válida, no entanto o sinal digitalizado não é um sinal IF, mas sim um sinal que provém de um detector de potência, portanto não será necessário aplicar a transformada de Fourier, pois o sinal lido através da ADC é um sinal que já contém a informação relativa á potência do sinal IF.

3.4.1 Estimador de potência

Para estimar o espectro, primariamente será utilizado uma Fast Fourier transform (FFT). Numa fase posterior será usado o método do periodograma de Welch.

O método consiste em dividir no tempo o sinal em P blocos de D amostras, podendo haver sobreposição de blocos. É aplicada aos blocos uma janela w(n) de modo que nas "pontas"de cada bloco o sinal seja zero e a reduzir o efeito da fase[20].

Cada bloco P é constituído por:

$$x^{(p)}(n) = w(n)x(n + PD); \tag{3.5}$$

O periodograma de o bloco p é:

$$\tilde{P}_{xx}^{(p)}(f) = \frac{1}{UDT} \left| X^{(p)}(f) \right|^2 \tag{3.6}$$

onde $X^{(p)}(f)$ é a FFT do sinal $x^{(p)}(n)$ e U é a energia da janela aplicada, que no caso de ser aplicada uma janela rectangular a energia é 1.

A média de todos os segmentos reside o periodograma de Welch, isto é:

$$\hat{P}_x(f) = \frac{1}{P} \sum_{p=0}^{P-1} \tilde{P}_{xx}^{(p)}(f) \tag{3.7}$$

3.4.2 Varrimento na frequência

É neste bloco que é feito o controlo do oscilador local e eventualmente o controlo de mais algum componente do front-end. No entanto o controlo do oscilador local é o que vai ser abordado neste capitulo.

Como já foi visto nas secções anteriores, devido a várias limitações o espectro terá de ser dividido em várias janelas de observação. É este bloco que se encarrega de deslocar a janela de observação ao longo do espectro radioeléctrico. A janela, como já foi referido atrás, terá de ter um deslocamento constante de modo a que o tratamento dos dados seja o mais simples possível.

Assumindo como anteriormente referido que os saltos no espectro têm de ser no máximo equivalentes à largura de banda ΔB imposta pelas limitações dos componentes do bloco downconverter e pela velocidade de transmissão de dados do bloco de comunicação, então o ΔB é o deslocamento em frequência do oscilador local.

Será necessário antes de mais definir a frequência na qual estará centrada a primeira janela de observação. Partindo desta frequência basta deslocar o oscilador ΔB para ficar centrado

na próxima janela de observação. Assim sendo pode-se dizer que a frequência central de cada janela pode ser dada por esta equação:

$$f_{c_i} = \Delta B \cdot i + f_{c_0}, \quad i = 0, 1, 2, \cdots, k-1 \tag{3.8}$$

onde f_{c_0} é a frequência central da primeira janela de observação e f_{c_i} é a frequência central da janela de observação i.

A frequência f_{c_0} por definição é a frequência central da primeira janela de observação. Como foi referido na secção anterior o início da primeira janela é colocado no início da gama do espectro a observar. O cálculo de f_{c_0} é dado pela seguinte fórmula:

$$f_{c_0} = f_{B_{ini}} + \frac{\Delta B}{2} \tag{3.9}$$

como foi definido anteriormente, $f_{B_{ini}}$ é o início da banda de observação. Nesta fórmula assume-se que o início da primeira janela de observação é colocado no início da banda de frequências a observar.

O controlo a partir deste dados é simples, basta colocar o oscilador local a uma frequência de modo a que as frequências centrais das bandas de observação passem para a frequência central de IF, isto é :

$$f_{LO} = f_{c_i} + f_{IF} \tag{3.10}$$

Está-se a assumir que f_{LO} tem de ser superior à frequência central da janela e que à saída de IF a imagem do sinal RF filtrada é a correspondente à subtracção, ou seja, a imagem centrada em $f_{RF} - f_{LO}$. A fórmula pode variar consoante o requisito para estes dois parâmetros, isto é, conforme a escolha da imagem do sinal RF a filtrar e conforme se o f_{LO} é superior ou inferior.

Na figura 3.11 é apresentado o espectro dividido em janelas de observação e as frequências centrais das janelas. Como se pode ver a $f_{B_{ini}}$ está distanciada da frequência central da primeira janela de observação de $\frac{\Delta B}{2}$.

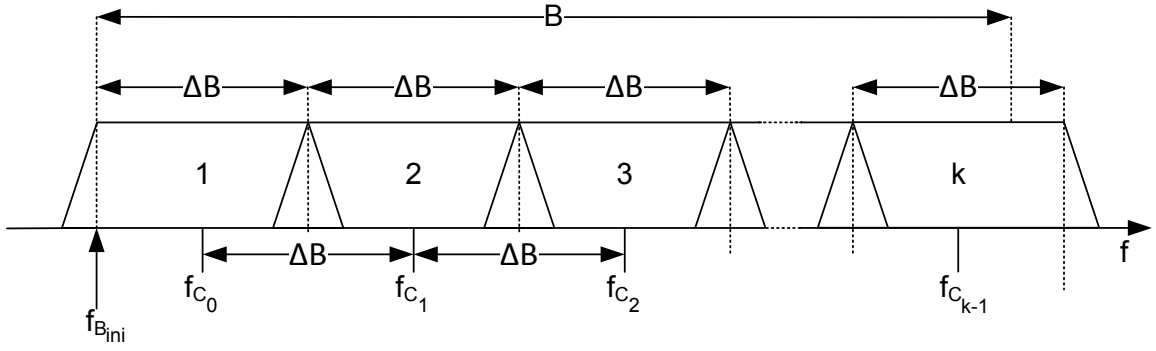


Figura 3.11: Divisão do espectro com as frequências centrais

Capítulo 4

Implementação do sistema

Neste capítulo vão ser apresentados os componentes constituintes do sistema e a sua implementação.

4.1 Escolha do controlador USB

O USB, tal como foi mencionado no capítulo anterior, será o protocolo usado para a comunicação entre o front-end e o PC. Será então importante escolher um controlador USB para o front-end que seja capaz de implementar este protocolo. O controlador de preferência deverá ter drivers já implementados.

Para a concretização deste projecto, foi proposto que se utilizasse o módulo UM245R da Future Technology Devices International Ltd. (FTDI) (figura 4.1) visto já ter sido utilizado anteriormente em projectos e ter uma fácil integração no sistema. Trata-se de um módulo de desenvolvimento que tem incorporado um chip FT245R que faz a conversão de USB para um interface paralelo First-in-First-out (FIFO) assíncrono. É um chip compatível com USB 2.0 full speed. Este chip tem implementados drivers para os sistemas operativos: Windows 7 de 32 e 64bits, Windows Xp e Xp 64-bit e ainda Windows Xp e Xp 64-bit.

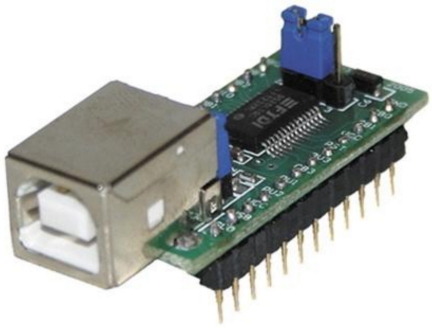


Figura 4.1: Módulo UM245R, retirado de [21]

No entanto as velocidades de transmissão de dados atingidas por este chip, visto que se trata de um USB 2.0 full speed, poderá não ser suficiente para transmitir os dados provenientes do front-end. Para contornar este possível problema tentou-se optar por um chip que implemente o USB 2.0 com a velocidade de High Speed.

Uma vez que a FTDI tem implementados alguns módulos de High Speed e tendo em conta que no passado tornou-se uma mais valia nos projectos desenvolvidos, a utilização destes módulos poderá ser benéfica.

Os módulos em questão são o FT2232H Mini Module e o FT4232H Mini Module. Apesar de terem uma aparência e funcionamento semelhante há uma diferença nos barramentos implementados. O módulo FT4232H Mini Module implementa a conversão de USB para barramentos série. É um módulo com 4 portos série capaz de atingir velocidades de 30Mbps. Tem incluído uma Electrically-Erasable Programmable Read-Only Memory (EEPROM) para guardar os dados relativos aos protocolos utilizados por cada porta série.

Contudo, apesar das suas características, apenas implementa barramentos série o que pode trazer problemas de velocidade para o interface com a ADC. Na próxima secção será apresentado o módulo FT2232H Mini Module que devido ás suas características se torna uma melhor opção.

4.2 FT2232H Mini Module

Desenvolvido pela FTDI, o FT2232H Mini Module (figura 4.2) é um módulo de desenvolvimento de USB 2.0 que faz a transição entre o barramento USB e barramentos série ou paralelos. Este módulo utiliza o chip de USB Hi-Speed, FT2232H, que contém dois portos que podem ser configurados para utilizar protocolos série síncronos e assíncronos ou o interface paralelo FIFO. Os dois portos ainda podem ser configurados independentemente para utilizar Multi-Portocol Synchronous Serial Engine (MPSSE). Isto permite que os portos do FT2232H sejam configurados para utilizar independentemente UART/Bit-Bang ou simular um dos protocolos suportados pelo MPSSE: Joint Test Action Group (JTAG), Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C), Bit-Bang entre outros.

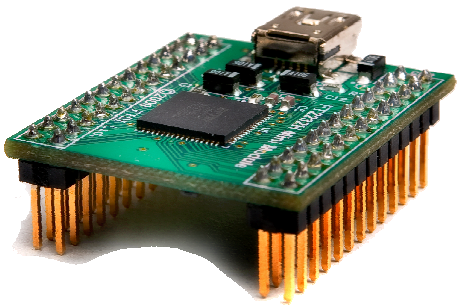


Figura 4.2: FT2232H Mini Module, retirado de [22]

Este módulo implementa o circuito básico para o funcionamento do chip FT2232H e foi desenvolvido para ser facilmente integrado em sistemas, colocando todos os pinos mais relevantes do chip à disposição do utilizador.

Nas próximas subsecções serão apresentadas, de forma mais detalhada, algumas características e funcionalidades deste módulo.

4.2.1 Características

O módulo, como já foi dito anteriormente, é uma placa que contém o circuito básico para o funcionamento correcto do chip FT2232H. As principais características que tornam este módulo tão atractivo são:

- Compatibilidade com o USB 2.0 e 1.1;
- Rápida integração em sistemas;
- Opção de poder ser alimentado por USB;
- Todo o protocolo USB é assegurado pelo módulo;
- Conector Mini USB do tipo B;
- Até 30Mbps nos protocolos JTAG, SPI, I^2C;
- 2 sockets de 26 pinos distribuídos por 2 filas com pitch de 2.54mm;
- Device drivers já implementados;
- EEPROM 93LC56 incluída;
- Regulador de 3V3.

4.2.2 Chip FT2232H

No módulo FT2232H Mini Module está incluído, como já foi referido, o chip FT2232H. Este chip é um controlador de USB e todo o protocolo é assegurado por este. Para além de assegurar o protocolo, este chip faz a ponte entre USB e outros barramentos, tanto série como paralelo. Os barramentos mais importantes e necessários para a implementação do sistema são: SPI, I^2C e FIFO síncrono. O SPI e o I^2C são barramentos série que podem atingir velocidades de transferência até 30Mbps, enquanto que o FIFO síncrono é um barramento paralelo que faz transferências sincronizadas a um relógio de 60MHz.

Na figura 4.3 pode-se ver o diagrama de blocos do chip. Como se verifica, o chip tem dois portos que são independentes e ambos têm associado a si um MPSSE, uma Universal asynchronous receiver/transmitter (UART) e um controlador FIFO que podem ter baud rates independentes. Os baud rates máximos podem variar dependendo do barramento em uso. Cada porto tem ainda agregado a si buffers de memória com 4KBytes para transmissão e recepção de dados.

Um dos blocos essenciais para o funcionamento correcto do barramento USB é o bloco Universal Transceiver Macrocell Interface (UTMI) PHY. Este tem a função de Serialise - Deserialise (SERDES) os dados recebidos e enviados pelo USB a Full-speed / High-Speed. Quem faz a gestão dos dados que vêm do UTMI para os portos é o bloco do USB Protocol Engine and FIFO control.

O bloco EEPROM Interface é outro dos blocos essenciais deste chip. Este permite, na presença de uma EEPROM, configurar os portos independentemente um do outro. Sem uma EEPROM não seria possível configurar os portos, e o chip apenas funcionaria como dois portos série.

4.2.3 Protocolos

Uma das grandes características deste módulo é a capacidade de fazer a ponte entre o barramento USB e os diversos barramentos. Nesta subsecção serão abordados apenas os barramentos a ser utilizados para o trabalho. Os barramentos a abordar serão o SPI e I^2C, ambos geridos pelo MPSSE, e o FIFO síncrono.

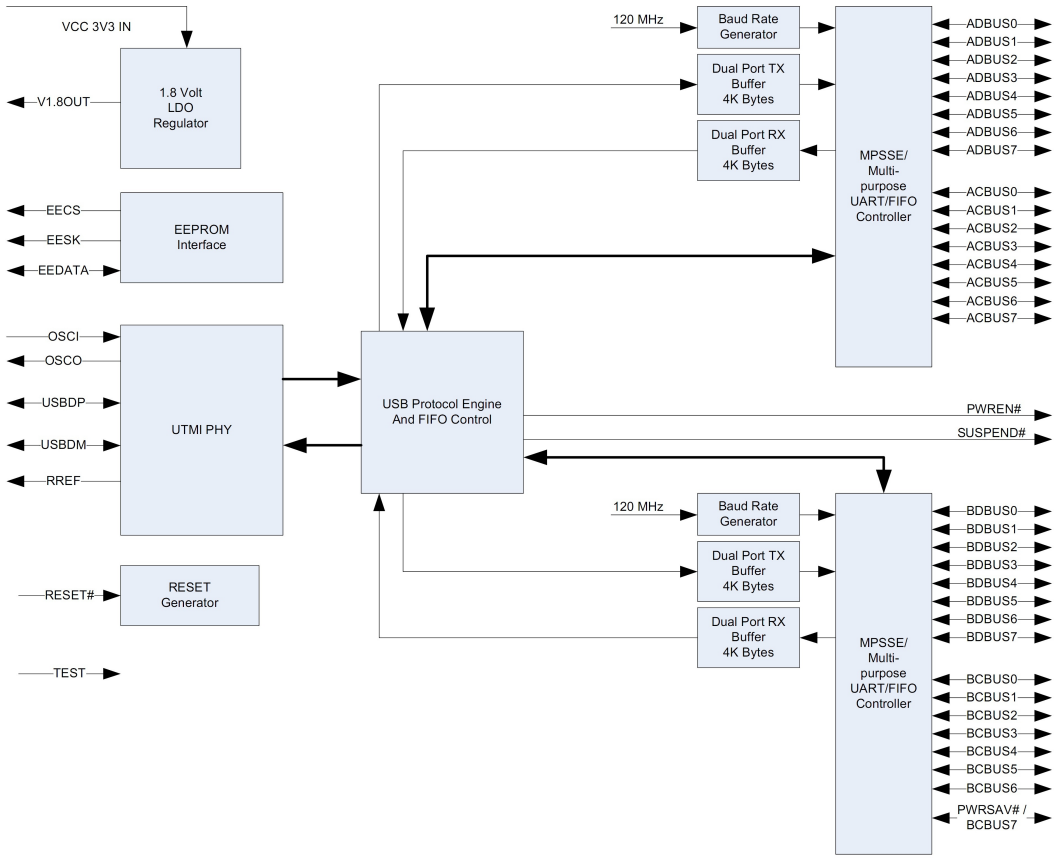


Figura 4.3: Diagrama de blocos FT2232H, retirado de [23]

SPI

O SPI é um barramento série síncrono que foi desenvolvido pela Motorola, e é constituído por um Master e por um ou vários Slaves [24].

Na figura 4.4 está o esquema genérico de um barramento SPI. Como se pode ver, o barramento é constituído apenas por 4 sinais:

- Um serial clock proveniente do Master;
- Uma entrada de dados, MISO, proveniente do Slave;
- Uma saída de dados, MOSI, proveniente do Master;
- Um chip select, proveniente do Master;

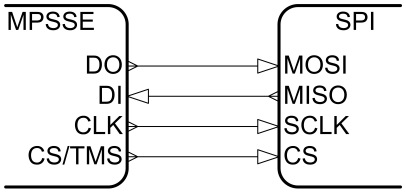


Figura 4.4: Barramento SPI com um Master e um Slave, retirado de [25]

No caso de haver mais que um Slave, o Master terá de ter mais que um chip select, pois a cada Slave corresponde um chip select como se pode ver na figura 4.5.

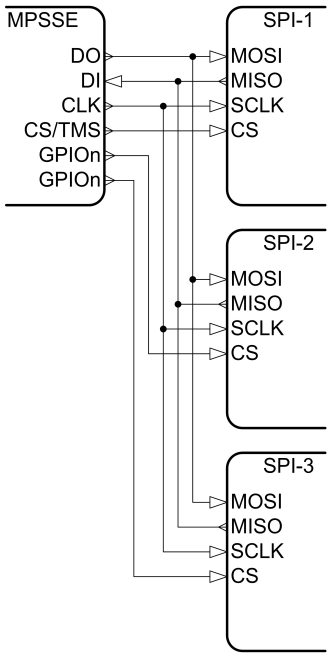


Figura 4.5: Barramento SPI com um Master e vários Slaves, retirado de [25]

A estrutura do barramento permite que sejam recebidos e enviados dados simultaneamente tornando-o um protocolo *full-duplex*. O funcionamento do barramento é muito simples: quando seleccionado o chip, através do chip select, o Slave poderá receber e/ou enviar dados à velocidade do clock fornecido, correspondendo a cada ciclo de relógio um bit recebido/enviado. Na figura 4.6 pode ver-se um tipo de transferência efectuada entre o Master e o Slave.

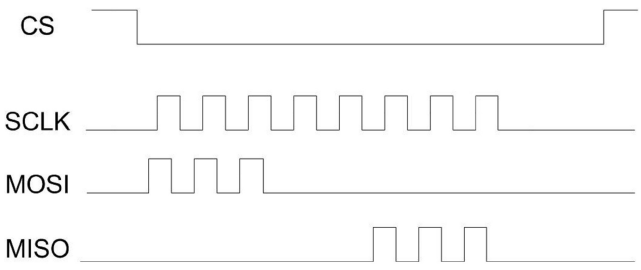


Figura 4.6: Diagramas temporais, retirado de [24]

O FT2232H Mini module é apenas capaz de implementar um controlador SPI *Master*. Para os vários slaves são utilizados os pinos GPIOn para fazer o chip select dos diferentes Slaves como se pode ver na figura 4.5. Este barramento no módulo pode atingir velocidades de 30Mbps.

I^{2}C

O barramento I^{2}C tem uma velocidade baixa/média de transferência de dados entre um master e um slave. Foi inventado pela Philips e é apenas constituído por dois fios, serial data, SDA, e serial clock, SCL, que transportam os dados entre dois dispositivos. Cada dispositivo tem um endereço único e pode trabalhar como receptor ou transmissor de dados. O master é o dispositivo que inicia a transferência de dados [26].

O I^{2}C é um verdadeiro barramento multi-master pois inclui um detector de colisões para evitar a corrupção de dados se dois ou mais Masters tentam simultaneamente iniciar uma transferência de dados. É um barramento série, orientado a 8 bits com transferências de dados bi-direccionais. É um barramento half-duplex, pois apenas pode ocorrer a transmissão de dados num sentido - só possui um sinal para dados.

O FT2232H Mini module apenas pode ser master e nunca slave. Como os dispositivos neste barramento têm um endereço único não é necessário chip select como no SPI. Tal como o barramento SPI, neste módulo também o barramento I^{2}C consegue atingir velocidades de 30Mbps.

Nas figuras 4.7 e 4.8 estão representados respectivamente, um barramento com um master e um slave e um barramento com master e vários slaves.

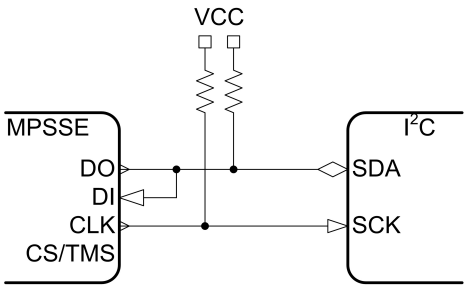


Figura 4.7: Barramento I^{2}C com um Master e um Slave[25]

É ainda apresentado o diagrama temporal de uma transferência na figura 4.9.

FT245 Style Synchronous FIFO

O FT245 Style Synchronous FIFO é mais um dos barramentos que é suportado pelo módulo. É um barramento paralelo síncrono, e faz as suas transferências de dados a um clock fixo de 60MHz. Na tabela 4.1 estão os sinais usados neste barramento e respectivas funções[27].

Para que este barramento funcione é necessário configurar a EEPROM de modo a que active este barramento. Uma das desvantagens de utilizar este barramento é que apenas o porto A está activo,deixando de funcionar o porto B, uma vez que todos os recursos do chip são utilizados no porto A.

Na figura 4.10 pode-se ver o diagrama temporal de uma leitura do FT2232H Mini Module(em cima) e de uma escrita para o mesmo(em baixo). Os requisitos temporais têm de ser cumpridos de modo a que seja possível a transferência de dados. Como já foi dito todas as transferências são feitas a um clock de 60MHz. Na tabela 4.2 estão expostos os tempos necessários para que este módulo funcione.

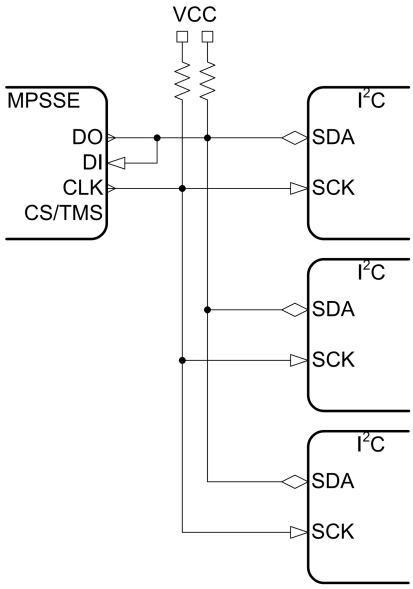


Figura 4.8: Barramento I^2C com um Master e muitos Slaves[25]

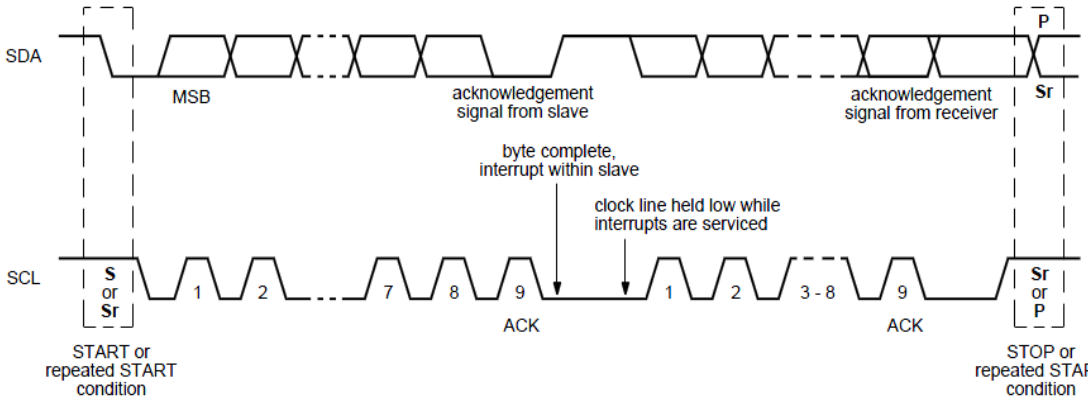


Figura 4.9: Diagrama temporal de uma transferência, retirado de [26]

Sinais	Tipo	Notas
ADBUS[7:0]	I/O	Dados FIFO bidireccionais. O barramento é input a não ser quando OE# está low
CLKOUT	OUTPUT	60MHz Clock provém do chip. Todos os sinais devem ser sincronizados com este clock.
RXF#	OUTPUT	Quando high, não lê dados do FIFO. Quando low, os dados do FIFO estão disponíveis e podem ser lidos forçando o sinal RD# a low.
OE#	INPUT	Permite o output dos dados para o chip. Deve ser colocado a low pelo menos 1 ciclo de relógio antes de colocar RD# low.
RD#	INPUT	Permite ao byte presente no momento na FIFO passar para o chip quando RD# está low.
TXE#	OUTPUT	Quando high, os dados não são escritos para o FIFO. Apenas é possível escrever para o FIFO quando este sinal estiver a low e o sinal WR# for puxado para low. Os dados são transferidos sempre que há uma transição ascendente do clock desde que TXE# e WR# estejam low
WR#	INPUT	Permite, quando low, que o byte presente no barramento seja escrito no FIFO. Até que WR# seja high em cada ciclo de relógio é transferido um byte para o FIFO

Tabela 4.1: Sinais usados no interface FT245 Style Synchronous FIFO

No entanto apesar de operar a 60MHz este barramento apenas é capaz de ler e escrever continuamente 510 Bytes, limitando assim a máxima capacidade de transmissão de dados.

Nome	Min	Nom	Max	Unidades	Notas
t1		16.67		ns	Período do CLKOUT
t2	7.5	8.33		ns	Período high do CLKOUT
t3	7.5	8.33		ns	Período low do CLKOUT
t4	1		7.15	ns	CLKOUT para RXF#
t5	1		7.15	ns	CLKOUT para dados validos
t6	1		7.15	ns	OE# para dados validos
t7	1		7.15	ns	CLKOUT para OE#
t8	11			ns	RD# setup time
t9	0			ns	RD# hold time
t10	1		7.15	ns	CLKOUT para OE#
t11	11			ns	Setup time da escrita de dados
t12	0			ns	Hold time da escrita de dados
t13	11			ns	WR# setup time
t14	0			ns	WR# hold time

Tabela 4.2: Requisitos temporais do modo FT245 Style Synchrounous FIFO

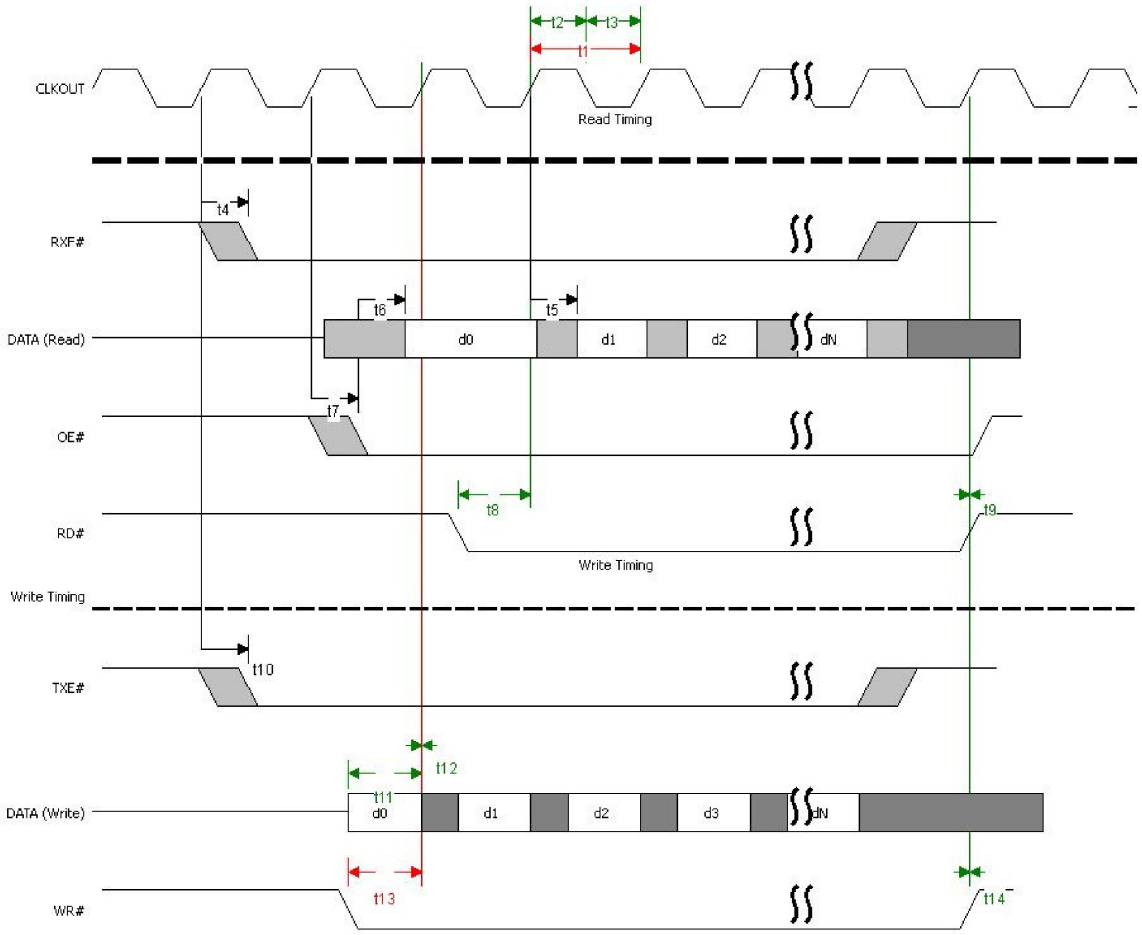


Figura 4.10: Diagrama temporal de leitura e escrita no modo FT245 Style Synchronous FIFO do FT2232H Mini module, retirado de [23]

4.3 Escolha da ADC

Na primeira fase desta dissertação, como se vai utilizar um detector de potência, não será necessário que a ADC tenha uma frequência de amostragem muito elevada mas sim uma grande precisão, visto que não será necessário fazer a FFT dos dados.

Para aumentar a precisão é necessário mais bits para codificar. Devido a que o interface FT245 Style Synchrounous FIFO é limitado a transferências de 8 bits, decidiu-se utilizar uma ADC que utilizasse um interface série para aumentar alguns bits na precisão. Na tabela 4.3 estão algumas ADCs que servem a este propósito.

ADC	Interface	Bits	Preço	f_a	SNR	SINAD
AD7621	SPI	16	53.03€	2MSPS	90dB	89.8dB
AD7623	SPI	16	40.04€	1.33MSPS	89.5dB	88.5dB
AD7476A	SPI	12	8.91€	1MSPS	71dB	70dB
ADCS7477	SPI	10	3.90€	1MSPS	62dB	61.7dB
ADC121S101	SPI	12	3.52€	1MSPS	72.5dB	72dB
ADCS7476	SPI	12	3.32€	1MSPS	72.5dB	72dB

Tabela 4.3: ADCs série

Como se pode ver na tabela 4.3 as ADCs a utilizar que mais se adquam a este trabalho são: a ADCS7476 e ADC121S101. Apesar das características de ambas serem iguais, diferem não só em preço mas também em consumo de potência. Assim sendo a melhor ADC é sem dúvida a ADC121S101.

Uma vez escolhida a ADC, a placa desenvolvida apenas deverá conter a ADC e o modulo USB. No anexo A.2 pode-se ver o esquema da placa desenvolvida.

Para se escolher a melhor ADC para a segunda fase desta dissertação, uma vez que se quer digitalizar um sinal IF, será escolhido o barramento que forneça a maior transferência de dados, pois quanto maior for a transferência de dados possível maior pode ser a frequência de amostragem. Como foi visto, o FT245 Style Synchrounous FIFO é o barramento que permite transferir dados a uma velocidade de 60MHz, portanto a ADC deverá atingir a velocidade deste barramento. O barramento fornece um clock que simplifica a tarefa e vai ser esta a frequência utilizada pela ADC para fazer a amostragem.

Depois de uma pesquisa de ADCs que obedecessem ao requisitos temporais impostos pelo barramento, não foi possível encontrar uma ADC que fornecesse os sinais para a escrita no módulo FT2232H Mini Module, no entanto, optou-se por uma ADC que apenas necessitasse de um sinal para a frequência de amostragem. Na tabela 4.4 estão todas as ADCs que apenas necessitam de uma frequência de amostragem.

ADCs	Effective bits	Power	Largura banda	SNR	Delay	Preço
ADC08L060	7.6 @39MHz	39mW	270MHz	47.2@29MHz	5	13.36 €
ADC08060	7.5 @25MHz	78mW	200MHz	46 @29MHz	2.5	7.91 €
ADC08100	7.2 @49.8MHz	78mW	200MHz	45.8 @45.8MHz	2.5	5.98 €
ADS830	7.7 @1MHz	170mW	300MHz	49.5 @10MHz	4	7.33 €

Tabela 4.4: ADCs

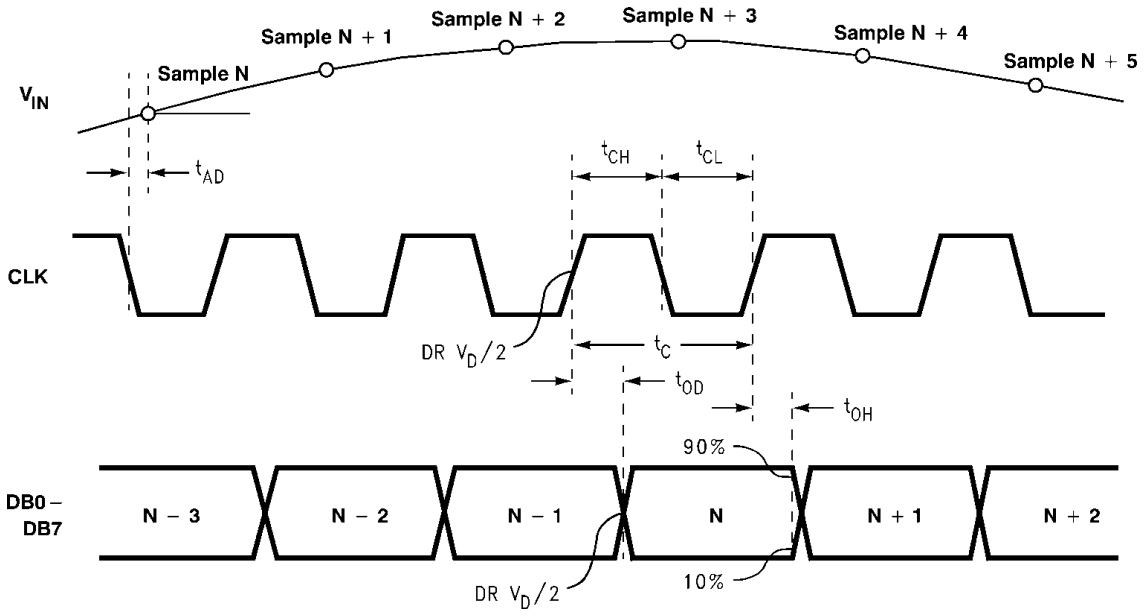


Figura 4.11: Diagrama temporal de uma conversão de uma ADC, retirado de [28]

Na figura 4.11 pode-se observar o diagrama temporal que é semelhante em todas estas ADCs. Se se comparar este diagrama temporal com o do barramento FT245 Style Synchrounous FIFO(figura 4.10) podem-se observar semelhanças no clock e barramento. No entanto, para o funcionamento correcto do barramento, será necessário ter uma lógica adicional, que será abordada na próxima secção.

Para a escolha da ADC será utilizado outro critério para além do interface. Comparando as ADCs, verificou-se que todas apresentam SNR, largura de banda e Effective bits bastante semelhantes. No entanto, diferem no pipeline delay, em consumo e em preço. A escolha mais viável, depois da análise destes factores, é a ADC08100, uma vez que apresenta um preço e um pipeline delay reduzidos.

Há apenas mais um aspecto a considerar em relação a estas ADCs: quando retirado o clock, o chip apenas consome 3mW. Quando esse é reposto, as amostras apenas serão válidas passado aproximadamente 1μs, o que corresponde a 60 amostras perdidas.

4.4 Máquina de estados e lógica adicional

Devido ao facto de a ADC não conseguir fornecer os sinais necessários para que o barramento FT245 Style Synchrounous FIFO funcione correctamente será preciso uma lógica adicional. A operação de leitura deste barramento utiliza os sinais TXE# e WR# tal como se pode ver na figura 4.10. Como o TXE# é um sinal fornecido pelo barramento, apenas será necessário fornecer um WR#. O problema pode ser resolvido através da construção de uma máquina de estados, pois são necessários sinais síncronos a um relógio. Para tal primeiro construiu-se o diagrama de estados e a tabela de estados. Como se pode ver pela tabela de estados(tabela 4.5) e pelo diagrama de estados (figura 4.12) existem dois estados possíveis. É portanto necessário apenas 1 bit para codificar os estados. Trata-se de uma máquina de

Moore pois a saída apenas depende do estado actual e não da entrada. Assim sendo, fazendo corresponder o estado ao par de saídas, isto é, A=1 e B=0, obtém-se a representação binária do estado. A tabela 4.6 representa a tabela de transições/ saídas, obtidas a partir das considerações feitas. A fase seguinte é escolher os Flip-Flops (FF) para a máquina de estados. Foi escolhido um FF do tipo D , que apresenta a seguinte equação característica $Q^* = D$ em que D é a entrada do FF do tipo D.

	Entrada TXE#		Saída WR#	
Estado	TXE#=0	TXE#=1	TXE#=0	TXE#=1
A	B	A	0	1
B	B	A	0	1

Tabela 4.5: Tabela de estados

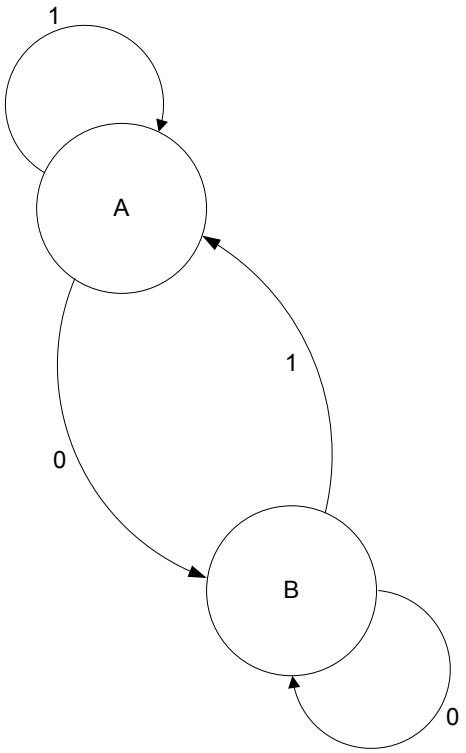


Figura 4.12: Diagrama de estados

A tabela 4.7 apresenta a tabela de excitação obtida através da substituição pelos FF. Estas tabelas também representam os mapas de Karnaugh.

Testou-se o circuito da máquina de estados apresentado na figura 4.13, obtido a partir do resultado da tabela de excitação. Como se pode ver o diagrama temporal deste circuito teórico é semelhante ao da figura 4.10, no que toca aos sinais TXE# e WD#.

Será então necessário escolher FF do tipo D que permita obedecer aos requisitos temporais necessários para que o barramento funcione correctamente. Na tabela 4.8 são apresentados alguns FF que obedecem aos requisitos mínimos.

	Entrada TXE#		Saída WR#	
Estado	TXE#=0	TXE#=1	TXE#=0	TXE#=1
1	0	1	0	1
0	0	1	0	1
Q_0	Q_0^*			

Tabela 4.6: Tabela de transições/saídas

	0	1
0	0	1
1	0	1

Q_0^*=TXE#

Tabela 4.7: Tabela de Excitação

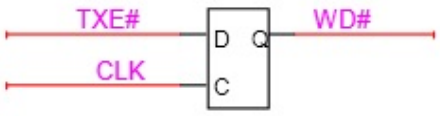


Figura 4.13: Máquina Estados

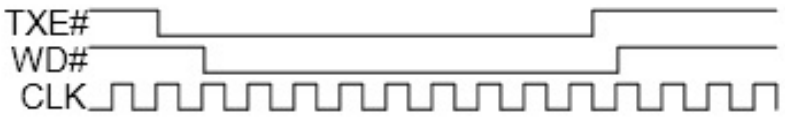


Figura 4.14: Diagrama temporal do circuito proposto

Flip-Flop	TIPO	Nº FF	$t_{su}(ns)$	$t_h(ns)$	$t_{pHL}(ns)$	Preço
SN74LVC74AD	D	2	3	0	5.2	0,44 €
SN74LVC2G74	D	1	1.3	1.2	5.9	0,52 €
SN74LVTH273	D	8	2.3	0	4.8	0,32 €
SN74ALVCH374	D	8	1.8	0.5	3.6	0,63 €
SN74LVC1G79	D	1	1.3	1	5	0,38 €

Tabela 4.8: Flip-Flops

Apesar de todos terem umas características muito semelhantes, o FF que fornece uma melhor solução é o SN74ALVCH374 visto que tem o menor tempo de setup, hold e de transição de high para low.

Será então necessário um FF SN74ALVCH374 para que o barramento funcione. No entanto, devido aos requisitos de corrente no barramento de dados, isto é, de D0 a D7, será necessário um FF deste tipo extra entre a ADC e o módulo FT2232H Mini Module de modo a que os dados possam ser lidos correctamente, apesar de ser imposta mais uma amostra de atraso.

4.5 Escolha do downconverter

Escolhido o controlador USB e a ADC, falta escolher o downconverter para finalizar o projecto do SDR. Na primeira parte desta dissertação não será necessário escolhê-lo, pois como já foi dito, este será desenvolvido no âmbito de outra dissertação de mestrado. No entanto será falado do sintetizador na próxima secção.

A ADC escolhida para a segunda parte da dissertação, como foi visto nas secções anteriores, trabalha a uma frequência de amostragem de 60MHz e tem uma largura de banda de 200MHz pelo que pode realizar subamostragem sem grandes problemas. Pode-se então escolher um IF que fique dentro da gama de amostragem ou nas gamas de subamostragem.

Apesar de haver uma vasta gama de conversores para IF integrados num só chip, apenas foi encontrado um que tem um módulo e obedece ao requisito de IF, pelo que se ficou muito limitado às características do mesmo. O módulo em questão é o MAX3542EVKIT.

4.5.1 Kit MAX3542EVKIT

Desenvolvido pela *Maxim Integrated Products* o kit MAX3542EVKIT simplifica os testes e a avaliação do chip MAX3542. Este é um tuner[1] de televisão que faz a conversão de uma gama de frequências contida na banda 47MHz a 862MHz para um IF de 36MHz. Este kit implementa o circuito básico para o funcionamento do chip MAX3542, e ainda colocado ao dispor do utilizador pinos de teste para verificar o funcionamento.

MAX3542

O chip MAX3542 implementa, como já foi dito, um tuner de televisão que faz a conversão de uma gama de frequências RF para IF. O sistema completo do chip inclui um Low Noise Amplifier (LNA) de ganho variável, um filtro variável programável, um mixer com harmonic-rejection, um LNA IF, um detector de potência IF e um amplificador de ganho variável IF. Inclui ainda 3 Voltage-Controlled Oscillators (VCOs), em que cada um tem 8 sub-bandas e uma Phase-Locked Loop (PLL). Na figura 4.15 pode-se ver o esquema geral do chip. Ambas, entrada e saida do kit, estão adaptadas a 50Ω.

Como se pode ver, estão incluídos no diagrama funcional dois divisores: o R e o N que fazem parte da PLL. Estes divisores são programáveis através de um interface série 2-wire compatível com I^2C. Para além de se poderem configurar estes divisores, é ainda possível programar diversos componentes do chip. Na figura 4.9 está uma tabela com os registos e os

[1] Um tuner é um circuito eléctrico que converte sinais rádio existentes no meio para sinais que possam ser processados posteriormente

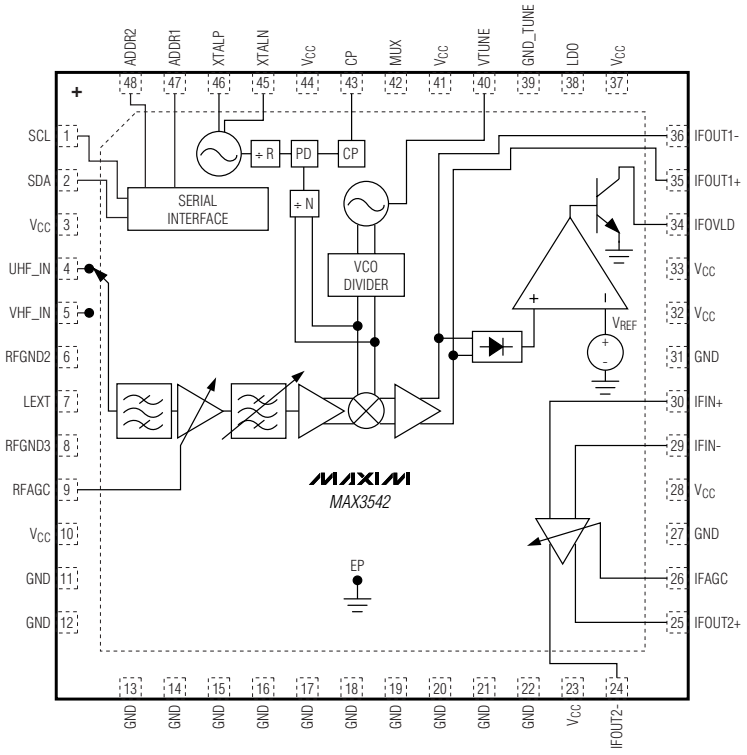


Figura 4.15: Diagrama funcional do MAX3542, retirado do [29]

endereços correspondentes, que são possíveis de programar ou de ler. Todos os registos têm 8 bits e no total há 13.

Os primeiros dois registos, destinam-se à programação do registo N. O terceiro registo destina-se a programação do divisor R. Devido às limitações das frequências sintetizáveis existem algumas restrições de valores nos registos N e R; o registo R pode variar entre 16 e 127 e o N pode variar entre 256 e 32,767. Para além destes divisores existe ainda um divisor à saída do VCO. Este é programável a partir do registo com o endereço 0x03 e pode tomar os valores de 4,8,16 e 32. Dependendo dos valores dos divisores uma frequência será sintetizada. No entanto não basta programar os registos, é preciso assegurar que a PLL atinge o lock, isto é, que a frequência desejada é sintetizada. Para isto é necessária uma escolha apropriada de um VCO e de uma sub-banda. Para verificar se foi atingido o lock, é necessário fazer uma leitura do registo Status, mais concretamente dos bits LD[2:0]. Se o valor em binário, destes bits estiver dentro da gama de 001 a 110, o lock da PLL foi atingido. Caso seja lido 111 a PLL não atingiu o lock , por isso deve de ser seleccionada uma sub-banda maior ou, se for o caso, um VCO que consiga sintetizar frequências maiores.. Se for lido 000 a PLL não terá atingido o lock e por isso deve de ser seleccionada uma sub-banda menor ou, se for o caso, um VCO que consiga sintetizar frequências menores.

Para além dos registos que contêm os divisores da PLL e divisor do VCO, existem também registos que permitem configurar o filtro variável. Este deve ser programado de forma a que seleccione a gama de frequências pretendida, de modo a optimizar a rejeição das interferências causadas pelo sinal fora de banda e minimizando o *insertion loss* para a gama de frequências desejadas. É através dos registo com o endereço 0x07 e 0x08, ou seja, registos Tracking Filter

REGISTER NAME	READ/ WRITE	REGISTER ADDRESS	MSB							LSB
			DATA BYTE							
			D7	D6	D5	D4	D3	D2	D1	D0
N-DIV High	Both	0x00	0	N14	N13	N12	N11	N10	N9	N8
N-DIV Low	Both	0x01	N7	N6	N5	N4	N3	N2	N1	N0
R-DIV	Both	0x02	0	R6	R5	R4	R3	R2	R1	R0
VCO	Both	0x03	VCO4	VCO3	VCO2	VCO1	VCO0	LD	VDIV1	VDIV0
IFOVLD, Charge Pump, and Filter Select	Both	0x04	0	IFOVLD2	IFOVLD1	IFOVLD0	CP1	CP0	TF1	TF0
Control	Both	0x05	0	0	0	0	SHDN _RF	SHDN _IFVGA	INPT1	INPT0
Shutdown	Both	0x06	SHDN _MIX1	SHDN _MIX0	SHDN _IF	SHDN _OD	SHDN _SYN	0	0	0
Tracking Filter Series Capacitor	Both	0x07	TFS7	TFS6	TFS5	TFS4	TFS3	TFS2	TFS1	TFS0
Tracking Filter Parallel Capacitor	Both	0x08	FLD	0	TFP5	TFP4	TFP3	TFP2	TFP1	TFP0
Tracking Filter ROM Address	Both	0x09	0	0	0	0	TFA3	TFA2	TFA1	TFA0
Reserved	Both	0x0A	X	X	X	X	X	X	X	X
ROM Table Data Readback	Read	0x0B	TFR7	TFR6	TFR5	TFR4	TFR3	TFR2	TFR1	TFR0
Status	Read	0x0C	POR	LD2	LD1	LD0	X	X	X	X

Tabela 4.9: Registos de Configuração, retirado do [29]

Series Capacitor e Tracking Filter Parallel Capacitor, mais concretamente os bits TFS[7:0] e TFP[5:0], que é possível alterar a banda de passagem do filtro variável. Os valores que devem ser escritos para estes registos são calculados a partir de fórmulas e devem ser alterados consoante a banda de frequências desejada. O MAX3542 inclui uma Read Only Memory (ROM) onde estão guardados valores que ajudam nestes cálculos. Cada chip é programado de fabrica contendo valores diferentes na ROM. Na tabela da figura 4.10 pode-se ver o conteúdo da ROM.

DESCRIPTION	ADDRESS	MSB							LSB
		DATA BYTE							
		D7	D6	D5	D4	D3	D2	D1	D0
Reserved	0x0	OD[2]	OD[1]	OD[0]	X	X	X	X	X
VHF Low	0x1	LS0[5]	LS0[4]	LS0[3]	LS0[2]	LS0[1]	LS0[0]	LS1[3]	LS1[2]
VHF Low	0x2	LS1[1]	LS1[0]	LP0[5]	LP0[4]	LP0[3]	LP0[2]	LP0[1]	LP0[0]
VHF Low VHF High	0x3	LP1[3]	LP1[2]	LP1[1]	LP1[0]	HS0[5]	HS0[4]	HS0[3]	HS0[2]
VHF High	0x4	HS0[1]	HS0[0]	HS1[3]	HS1[2]	HS1[1]	HS1[0]	HP0[5]	HP0[4]
VHF High	0x5	HP0[3]	HP0[2]	HP0[1]	HP0[0]	HP1[3]	HP1[2]	HP1[1]	HP1[0]
UHF	0x6	US0[5]	US0[4]	US0[3]	US0[2]	US0[1]	US0[0]	US1[5]	US1[4]
UHF	0x7	US1[3]	US1[2]	US1[1]	US1[0]	UP0[5]	UP0[4]	UP0[3]	UP0[2]
UHF	0x8	UP0[1]	UP0[0]	UP1[5]	UP1[4]	UP1[3]	UP1[2]	UP1[1]	UP1[0]

Tabela 4.10: Tabela ROM, retirado de [29]

Os valores óptimos do filtro são dados pelas seguintes formulas:

- VHF LOW:

$$TFS = INT[10^{[(1.1\times\frac{LS0}{64}+2.2)+(4\times\frac{LS1}{16}-12)\times f_{RF}\times 10^{-3}]}] - 10 \quad (4.1)$$

$$TFP = INT[10^{[(0.8\times\frac{LP0}{64}+1.6)+(8\times\frac{LP1}{16}-14)\times f_{RF}\times 10^{-3}]}] \quad (4.2)$$

- VHF HIGH:

$$TFS = INT[10^{[(1.3\times\frac{HS0}{64}+2.5)+(4\times\frac{HS1}{16}-8)\times f_{RF}\times 10^{-3}]}] - 10 \quad (4.3)$$

$$TFP = INT[10^{[(0.8\times\frac{HP0}{64}+1.6)+(1.6\times\frac{HP1}{16}-3.2)\times f_{RF}\times 10^{-3}]}] \quad (4.4)$$

- VHF HIGH:

$$TFS = INT[10^{[(\frac{US0}{64}+3)+(2\times\frac{US1}{64}-3)\times f_{RF}\times 10^{-3}]}] - 20 \quad (4.5)$$

$$TFP = INT[10^{[(0.8\times\frac{UP0}{64}+1.6)+(1.6\times\frac{UP1}{64}-2.5)\times f_{RF}\times 10^{-3}]}] - 10 \quad (4.6)$$

onde:

f_{RF} = A frequência de operação em MHz;

TFS = Valor decimal do valor óptimo de TFS[7:0] , para uma dada frequência de operação;

TFP = Valor decimal do valor óptimo de TFP[5:0] , para uma dada frequência de operação;

$LS0$, $LS1$, $LP0$, $LP1$, $HS0$, $HS1$, $HP0$, $HP1$, $US0$, $US1$, $UP0$ e $UP1$ = valor decimal dos coeficientes da tabela ROM.

Diagrama funcional

Referiu-se anteriormente que o sistema inclui um chip MAX3542. No entanto para o funcionamento correcto do chip tem que se implementar o circuito básico. O Kit já inclui este circuito.

Para além disso, este circuito delimita o sinal à entrada de modo a que pertença à gama que opera o circuito e implementa filtros suplementares. Na figura 4.16 pode-se observar o sistema implementado pelo kit MAX3542EVKIT. No entanto, a saída diferencial IFOUT no kit é transformada numa saída single-ended, através de uma bobina.

Antes de IFOUT existe um filtro adicional anti-aliasing de modo a ser possível a digitalização do sinal. Ainda foi acrescentado ao sistema um outro filtro. Este filtro tem uma resposta em frequência igual à apresentada na figura 4.17. Como se pode ver, o filtro está centrado em 36MHz delimitando o sinal que passa a uma banda de passagem com uma largura de 8MHz. Portanto, o sinal IF tem uma largura de banda de 8MHz e está centrado em 36MHz.

À entrada do sistema, existe um diplexer separando as frequências pertencentes a UHF das frequências pertencentes a VHF. Este diplexer delimita também a largura de banda do sinal à entrada.

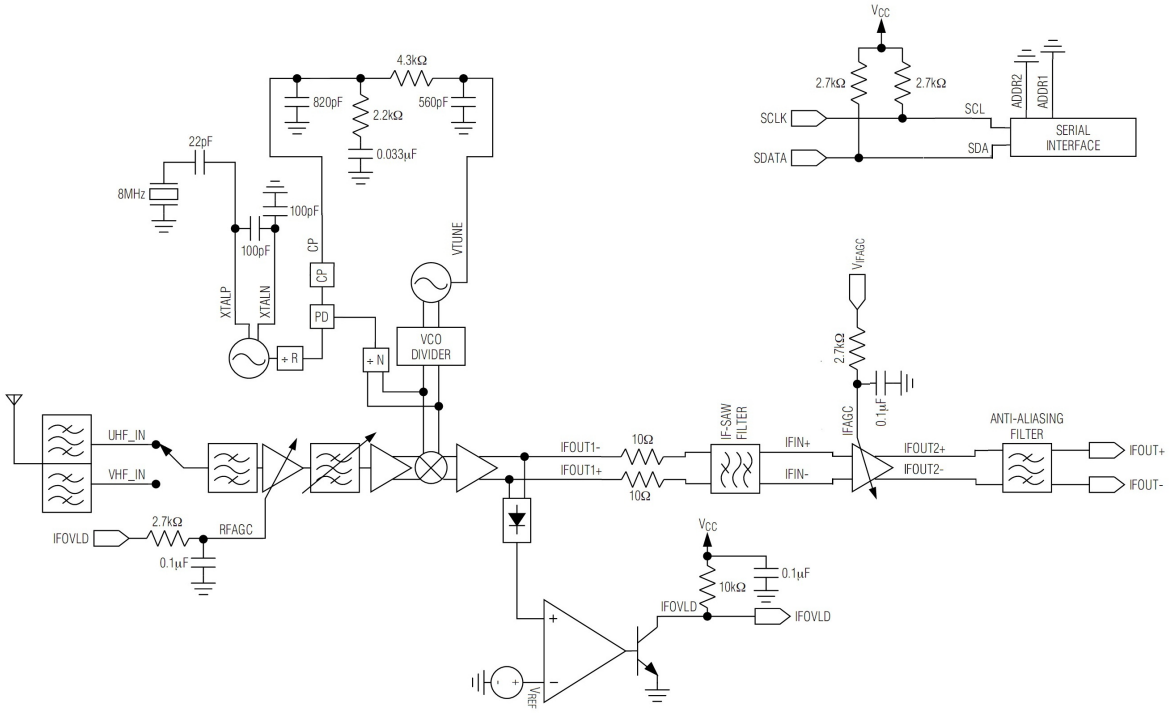


Figura 4.16: Diagrama funcional do kit MAX3542EVKIT, adaptado de [29]

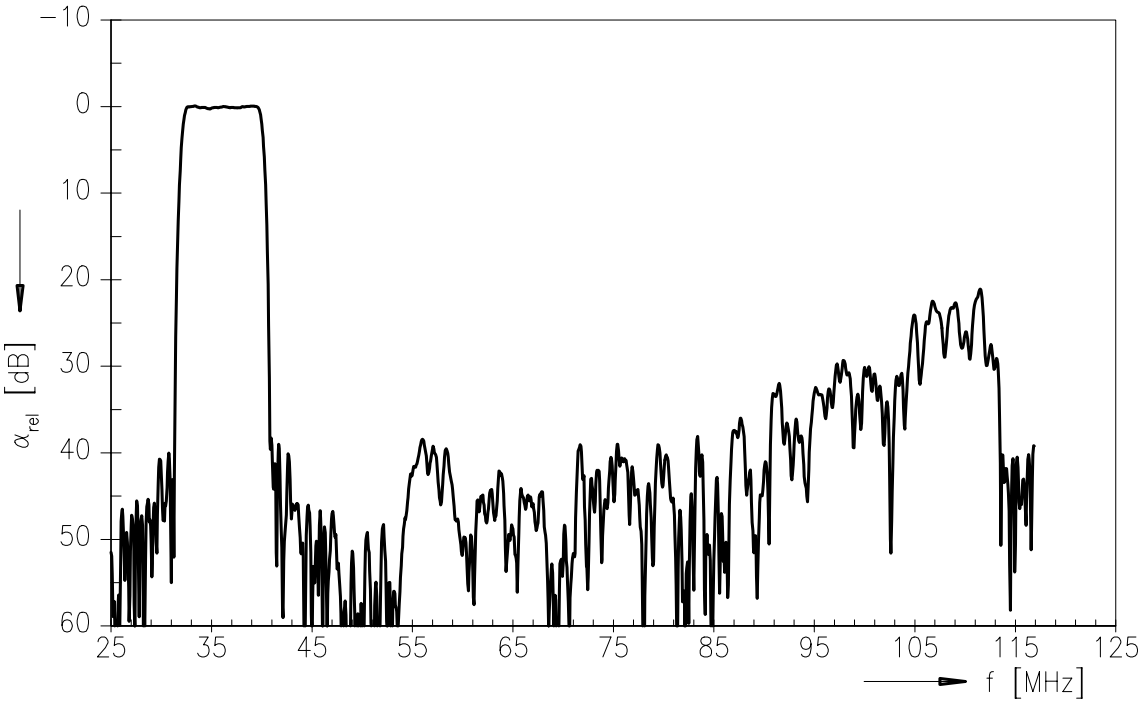


Figura 4.17: Resposta em frequência do filtro, retirado de [30]

4.6 Placa de Desenvolvimento para o MAX3542EVKIT

Nas secções anteriores foram apresentados de um modo geral todos os componentes importantes do sistema. O kit MAX3542EVKIT, como já foi visto, precisa de ser ligado a ADC e a ADC ao controlador USB, tal como se pode ver na figura 4.18. Será necessário desenvolver uma placa que permita a ligação entre estes diversos componentes do sistema.

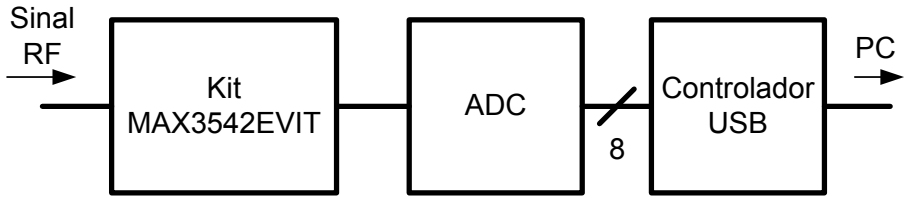


Figura 4.18: Esquema de ligação KIT MAX3542EVKIT, ADC e Controlador USB

Como o kit MAX3542EVKIT já vem adaptado a 50Ω com conectores SMA[2], basta criar um placa que faça a ligação a este kit através de SMA. Criou-se então uma placa com a ADC e com o controlador USB. No anexo A, mais concretamente na secção A.1, podem-se ver os esquemas da placa e o layout da PCB. Criou-se esta placa com o intuito de se tornarem mais fáceis os testes ao sistema. Devido ao encapsulamento dos componentes pertencentes ao sistema não foi possível desenvolvê-lo numa placa branca. Optou-se por criar uma placa que implementasse a ligação entre a ADC e o controladorUSB. O esquema eléctrico e o Printed circuit board (PCB) da placa foram desenvolvidos no EAGLE[3] Layout Editor, programa da CadSoft Computer.

As principais características desta placa de desenvolvimento são:

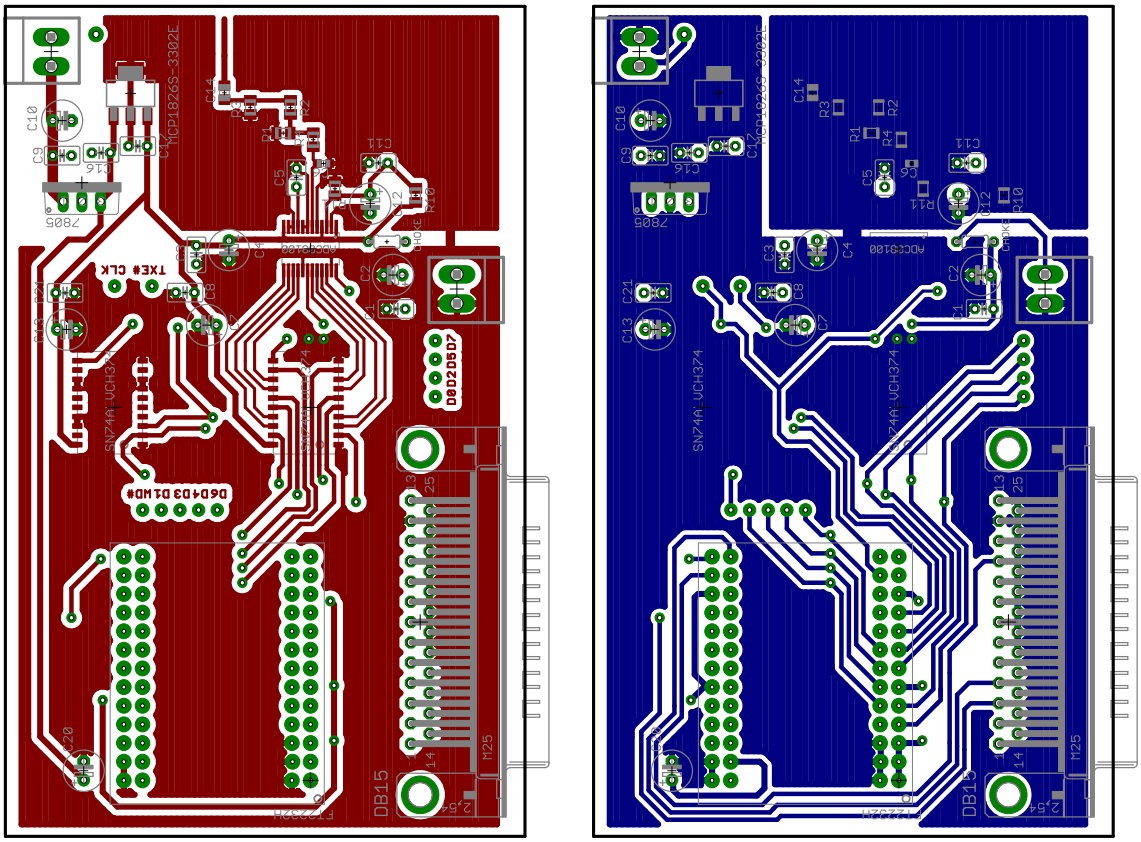
- ADC08100;
- Circuito básico para o funcionamento da ADC08100;
- Módulo FT2232H Mini module;
- Regulador 3.3V Low Dropout
- Regulador 5V Low Dropout
- Conector SMA para interface com o Kit MAX3542EVKIT
- Conector DB15 para interface com o Kit MAX3542EVKIT
- Pinos de alimentação 3.3V e Ground para o o Kit MAX3542EVKIT
- Tensão de Alimentação 10V
- Pinos de teste do barramento do controlador USB

Desde o desenvolvimento de um protótipo até à versão final do *layout* tiveram-se em conta diversas considerações de modo a optimizar a disposição dos componentes para que o comprimento das pistas fosse minimizado. No entanto, como o controlador de USB dispõe os seus sinais em pinos não foi possível diminuir muito o tamanho do layout da placa. Como também são necessários pinos de teste, o espaço ocupado por estes não ajudou na redução do tamanho da placa.

Quanto à alimentação da placa é assegurada por dois reguladores um de 5V e outro de 3.3V. Como é necessário alimentar o controlador USB, a ADC, o kit MAX3542EVKIT e os FF será preciso ter em conta a corrente necessária para que o sistema funcione. O controlador USB

[2]SMA ou SubMiniature version A são conectores coaxiais de RF

[3]O nome EAGLE é um acrónimo, que representa : Easily Applicable Graphical Layout Editor



(a) TOP (b) BOTTOM

Figura 4.19: PCB

consome no máximo 200mA, o kit no máximo consome 500mA, a ADC consome no máximo 100mA e os FF em conjunto consomem no máximo 300mA. No total será necessário um regulador dos 5V que proporcione uma corrente de 1.1A portanto foi escolhido um regulador capaz de fornecer 1.5A. No entanto nem todos os componentes necessitam de uma tensão de 3.3V.Portanto, é apenas necessário para estes componentes um total de corrente de 0.9A, pelo que um regulador de 3V3 capaz de fornecer 1A é o suficiente. Para regulador de 5V escolheu-se um regulador da família 7805 o L7805CV da STMICROELECTRONICS e para o regulador de 3.3V o MCP1826S-3302E/DB da MICROCHIP.

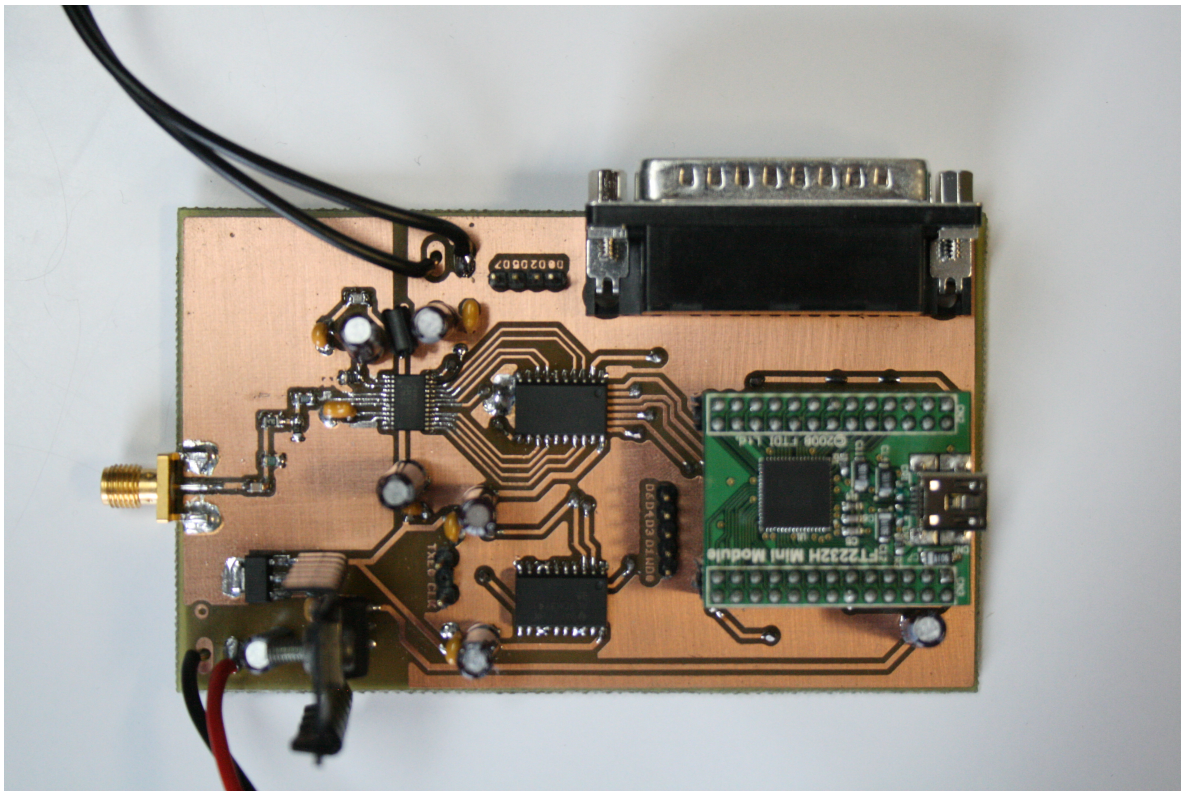


Figura 4.20: Placa final

Capítulo 5

Interface gráfica e controlo do analisador de espectros

5.1 Introdução

No próximo capítulo serão apresentadas a linguagem de programação, as bibliotecas usadas no trabalho, mas também como é feito o controlo de amostragem, o controlo do kit MAX3542EVKIT e o algoritmo de controlo.

5.2 Linguagens e Bibliotecas

5.2.1 Bibliotecas FTDI

Existem duas maneiras de interagir com o controlador USB. Uma é através dos drivers Virtual COM port (VCP) que fazem com que o USB simule uma porta COM[1]. O software desenvolvido pelo utilizador pode fazer o acesso ao USB da mesma maneira como acedia a uma porta COM standard. No entanto apesar de se poder fazer uma interacção simples não é possível configurar a EEPROM de modo a que o controlador USB funcione em outros modos. Será então necessário utilizar o outro tipo de drivers.

D2xx drivers

Uma outra maneira de fazer o interface com o USB é através dos drivers D2XX. A aplicação pode interagir com o módulo USB através de uma série de funções chamadas a partir de um Dynamic-link library (DLL) proprietário(FTD2XX.DLL). O interface D2XX oferece funções especiais que não estão disponíveis nas portas COM standards dos sistemas operativos, tal como a que já foi referida: mudar o modo em que o controlador está a funcionar.

Apesar das diferenças, os drivers para o Windows funcionam sobre a mesma camada de drivers, chamada Combined Driver Model (CDM). Na figura 5.1 pode-se ver a arquitectura do Windows CDM Driver.

Este drivers podem ser chamados a partir de diversas linguagens tais como C++, C#, Delphi, LabVIEW, Visual Basic, Java, Visual C++ entre outras. Portanto a escolha é limitada a estas linguagens.

[1]Uma porta COM é o nome original para uma porta série.

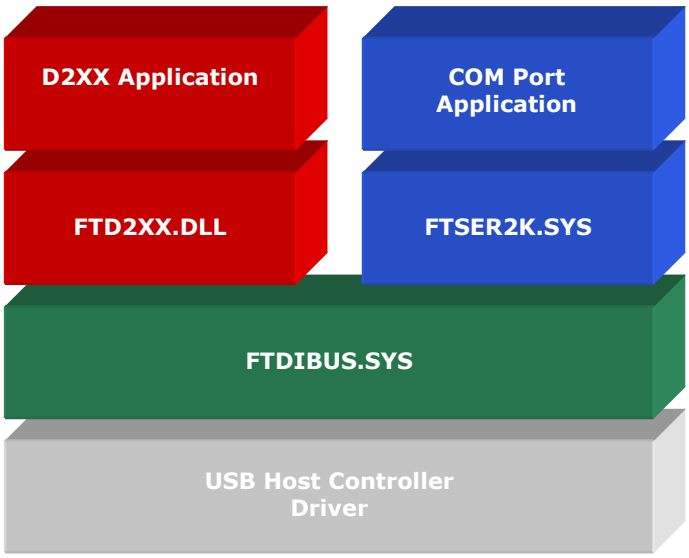


Figura 5.1: Arquitectura do Windows CDM Driver, retirada de [31]

Existem bibliotecas implementadas a partir dos drivers *D2XX* que implantam os barramentos série gerados pelo MPSSE. Nas próximas subsecções serão falados brevemente destes drivers.

MPSSE I2C DLL

O MPSSE I2C DLL é uma biblioteca que permite controlar o barramento I^2C. Para utilizar esta biblioteca é necessário que estejam a ser utilizados os drivers D2XX. Esta biblioteca simplifica o interface com o barramento I^2C de modo a que o utilizador não tenha que a implementar a partir do DLL dos drivers D2XX.

As linguagens suportadas por esta biblioteca são: C#, Delphi, Visual Basic 6, Visual Basic .NET, Visual C++ .NET.

MPSSE SPI DLL

Tal como a biblioteca MPSSE I2C DLL, a biblioteca MPSSE SPI DLL fornece ao utilizador um interface simples com o barramento SPI de modo a que este não os tenha que implementar a partir da biblioteca dos drivers D2XX. Como já foi dito é necessário ter instalados os drivers D2XX para que a biblioteca funcione correctamente.

Esta biblioteca é suportada pelas linguagens: C#, Delphi, Visual Basic 6, Visual Basic .NET, Visual C++ .NET.

Escolha da linguagem

Para que a implementação do sistema seja possível é necessário garantir que a linguagem consiga utilizar as bibliotecas MPSSE I2C DLL e MPSSE SPI DLL e fazer o interface com os drivers D2XX. Escolheu-se então uma linguagem comum às três bibliotecas e que a sua implementação se torna o mais simples possível. Para isto utilizou-se a linguagem Visual C++. Na próxima subsecção será abordada esta linguagem.

5.2.2 Visual C++/CLI

A linguagem C++/CLI é uma linguagem de programação standard criada pela Microsoft. É uma linguagem que faz a ligação entre o C++ Standart e a Common Language Infrastructure (CLI)[2]. Esta linguagem é suportada pela .NET Framework visto que se tratar do CLI.

Microsoft .NET Framework 3.5

Segundo a Microsoft o .NET Framework é um ambiente de desenvolvimento e execução que permite que diferentes linguagens de programação e bibliotecas funcionem em conjunto para criar aplicações baseadas em Windows que são facilmente compiladas, geridas, implementadas e integradas com outros sistemas baseados em rede.

Apesar de ser uma plataforma orientada a aplicações interligadas pela rede, em que parte do sistema pode ser corrido em diferentes partes do globo, a plataforma permite desenvolver, implementar e executar Serviços Web, aplicações Web, aplicações gráficas e ainda aplicações de consola. Todas estas peças podem ajudar ao desenvolvimento deste sistema.

O pretendido é construir um interface gráfico que seja fácil de desenvolver, portanto esta plataforma é uma mais valia, visto ser possível desenvolver aplicações gráficas. Estas utilizam uma biblioteca chamada *Windows Forms*. Esta biblioteca implementa facilmente interfaces gráficos, com a mesma facilidade que em *Visual Basic*. Quando são criadas *Windows Forms* vai ser usado o enorme namespace `System::Windows::Forms`.

O .NET Framework providencia um Graphical Device Interface (GDI) chamado GDI+ que permite mudar as fontes, as cores e ainda desenhar figuras. Este GDI+ permite criar imagens que podem ser uma mais valia quando se quiser guardar dados de medições. Para utilizar esta biblioteca será necessário utilizar namespace `System::Drawing`.

5.2.3 NPLOT

Como será necessário fazer uma representação gráfica do espectro será necessária uma biblioteca que tenha esta funcionalidade. Para este fim foi escolhida a biblioteca NPLOT que é uma biblioteca livre para fazer plots de dados para a plataforma .NET Framework. O NPLOT inclui controlos para a Windows.Form, ASP.NET e uma classe para criar Bitmaps e possui uma elegante e flexível Application Programming Interface (API) [32].

Esta biblioteca é compatível com o .Net Framework 1.1 e 2.0 o que o torna compatível com Visual C++/CLI. Na figura 5.2 pode-se ver um exemplo de um plot realizado por esta biblioteca.

No anexo B foi feito um pequeno tutorial de como utilizar esta biblioteca no Visual C++ 2008 - Express Edition.

5.2.4 FFTW

Na segunda parte desta dissertação será necessário fazer a FFT dos dados recebidos. Para tal vai ser utilizada a biblioteca FFTW

[2]Common Languague Infrastruture é um ambiente onde várias linguagens de programação, compatíveis com este ambiente, são compilados para uma linguagem neutra chamada Common Intermediate Language (CIL) sem a necessidade de ser reescritas. Esta linguagem CLI é compilada para código máquina através de Common Language Runtime (CLR). Esta CLI é uma especificação aberta desenvolvida pela Microsoft.

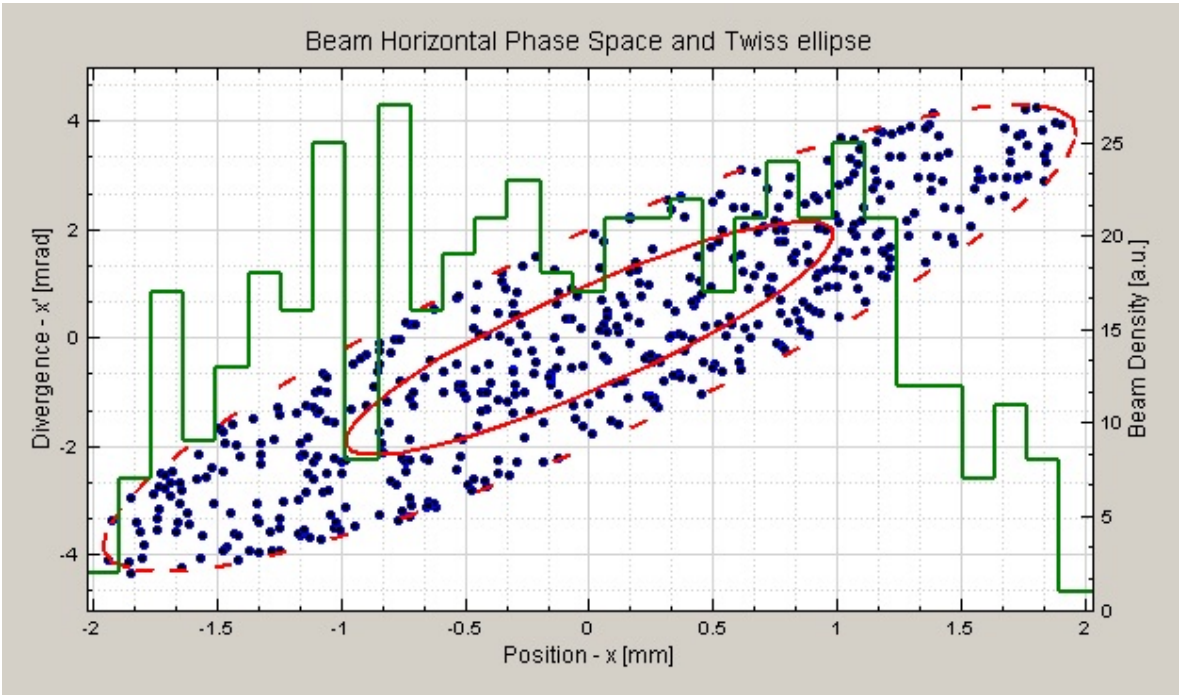


Figura 5.2: Exemplo de um gráfico utilizando o NPLOT [32]

A FFTW é uma biblioteca C de funções para processar a Discrete Fourier transform (DFT) em uma ou mais dimensões, com uma entrada de tamanho arbitrário, para dados reais e complexos. Foi desenvolvido pelo MIT[3] e é uma biblioteca livre.

A última versão é a FFTW 3.2.2 e as suas características são:

- Velocidade;
- Transformadas de uma ou multi dimensão
- Transformadas com um tamanho arbitrário
- Transformadas rápidas para dados de entrada e saída puramente reais;
- Paralelização das transformadas
- É um software livre, com licença GNU General Public License.

Modo de utilização

Para utilizar esta biblioteca é necessário primeiro criar um plano colocando os dados para calcular a FFT num buffer *in. Depois tem de se chamar a seguinte função para criar o plano:

```
fftw_plan fftw_plan_dft_1d(int n, fftw_complex *in, fftw_complex *out,
                                  int sign, unsigned flags);
```

Depois de chamada esta função é necessário que seja aplicada a transformada aos dados. Para tal será necessário chamar a seguinte função:

```
fftw_execute(p);
```

Depois de executada o resultado será exposto no buffer de saída *out.

[3]Massachusetts Institute of Technology é uma universidade de pesquisa privada localizada nos Estados Unidos da América, Cambringe.

5.3 Interface Gráfico e implementação

Construiu-se a partir da linguagem Visual C++/CLI um interface gráfico. Como já foi referido utilizou-se a biblioteca NPLOT para fazer o plot do espectro. Na figura 5.3 está exposto o interface criado para esta dissertação. Este interface permite guardar e imprimir o espectro. Para além disso permite aumentar a zona de observação. No anexo C está o manual de utilização do interface.

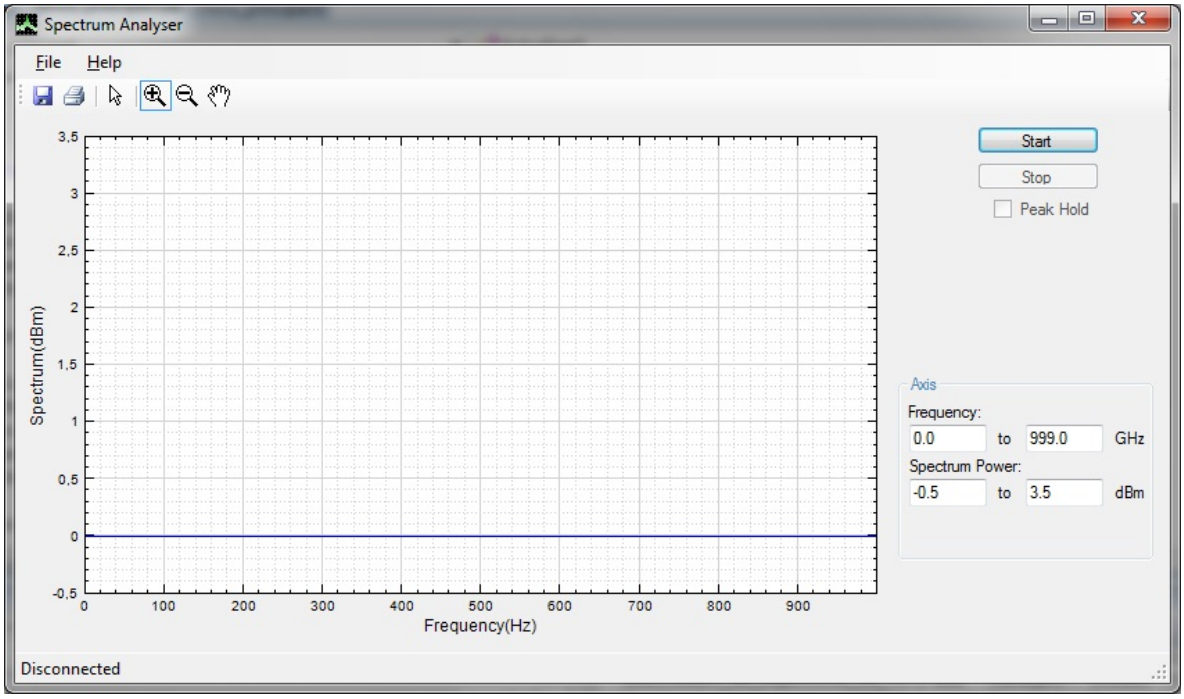


Figura 5.3: Interface gráfico

Para além deste interface gráfico foram criadas algumas bibliotecas que permitem simplificar o uso dos drivers do módulo FT2232H Mini Module. Foi criado a partir dos drivers D2XX a biblioteca FTDI_FIFO, e das bibliotecas MPSSE I2C DLL e MPSSE SPI DLL foram criadas as bibliotecas FTDI_NET_I2C e FTDI_NET_SPI, respectivamente.

Devido ao facto de a biblioteca FFTW ser uma biblioteca em C a conversão para C++/CLI não é fácil. Como a linguagem C++/CLI é uma linguagem com objectos *managed* a biblioteca tem de ser transformada neste tipo. Para isto é necessário que se façam primeiro um DLL que chame a biblioteca FFTW. Esta biblioteca tem ser escrita em C.

De seguida é necessário criar uma classe que utilize a biblioteca criada no DLL, referenciando estas classes em *managed* class. A partir de esta classe pode ser utilizada a biblioteca FFTW.

No esquema da figura 5.4 pode-se ver como é chamada a biblioteca FFTW. A classe FFTW_Managed é a biblioteca implementada no ambiente Visual C++, ou seja, é uma biblioteca *managed*. Como se pode ver esta biblioteca chama outra biblioteca que é escrita em C++ e não é *managed* que é a FFTW_Unmanaged. Nesta última é então chamada a biblioteca FFTW.

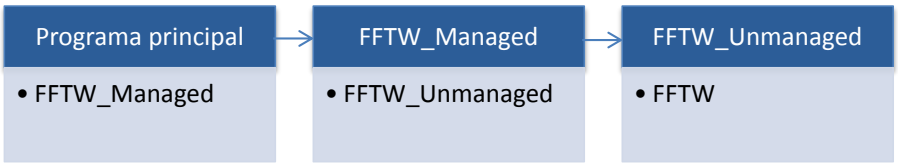


Figura 5.4: Esquema de utilização da biblioteca FFTW

5.4 Controlo do sintetizador

Na primeira fase desta dissertação,onde irá ser utilizada o front-end criado no âmbito de outra dissertação, é necessário controlar a frequência a sintetizar de modo a que seja possível o funcionamento correcto do sistema. Para tal é necessário controlar qual o oscilador a utilizar e controlar os seus registos N e R associados. Para controlar qual o oscilador apenas é necessário programar o registo N do oscilador a programar. Uma vez escolhido o oscilador é necessário sintetizar a frequência.

Para que seja possível sintetizar uma frequência é necessário que as frequências à entrada do detector de fase sejam iguais para isso é necessário que não só:

$$f_{PDx} = \frac{f_{XIN}}{R_{RFx}} \tag{5.1}$$

mas também:

$$f_{PDx} = \frac{f_{RFOUT}}{2 \times N_{RFx}} \tag{5.2}$$

em que R_{RFx} e N_{RFx} representam os divisores R e N consoante se está a trabalhar na frequência RF1 ou RF2. f_{PDx} é a frequência à entrada do detector de fase para cada banda. f_{XIN} é a frequência de referência e f_{RFOUT} é a frequência sintetizada. De modo a que seja sintetizada a frequência desejada é necessário que para a mesma banda de frequências a f_{PDx} seja igual. Deverá primeiro escolher-se uma frequência de f_{PDx}, consoante a precisão a que deve de ser deslocado o espectro, e escolheu-se portanto uma frequência de 100kHz, uma vez que fica dentro dos limites apresentados pelo datasheet do sintetizador. Uma vez escolhido é já possível escolher o R_{RFx}, que irá ser igual para as duas gamas. Sabendo que a frequência f_{XIN} é 10MHz, a partir da fórmula 5.1, é calculado o valor de R que é 100.

É possível desenvolver uma fórmula para facilitar o cálculo de N_{RFx}. Substituindo f_{PDx} por 100kHz na equação 5.2 obteve-se a seguinte equação:

$$N_{RFx} = \frac{f_{RFOUT}}{200kHz} \tag{5.3}$$

Partindo da equação 5.3 é possível calcular rapidamente o Nx.

Uma vez escolhidos os valores de f_{RFOUT} poder-se-ão calcular os valores que Nx deve de tomar. Não será necessário assegurar que o sintetizador atinja o lock pois este processo é automático. Uma vez calculados os valores tanto de N como de R poder-se-ão programar os respectivos registos do sintetizador de modo a que as frequências desejadas sejam sintetizadas.

5.5 Controlo do módulo MAX3542EVKIT

Na segunda parte desta dissertação, para que o sistema funcione é necessário controlar o oscilador que pertence ao kit. Para controlar é necessário programar os registos N e R que controlam a frequência sintetizada. Para isso primeiro é necessário compreender o esquema do sintetizador. Na figura 5.5 pode-se ver o esquema simplificado do sintetizador. Pode-se ver que no Phase Detector (PD) entram dois sinais. De modo a que uma frequência seja sintetizada correctamente estes dois sinais devem ter uma frequência igual. Numa entrada está um sinal com uma frequência:

$$f_{PD} = \frac{f_{REF}}{R} \tag{5.4}$$

enquanto que noutra entrada está um sinal com uma frequência igual a:

$$f_{PD} = \frac{f_o}{N} \tag{5.5}$$

onde f_{REF} é a frequência de referência que neste caso corresponde a 8MHz, f_o é a frequência sintetizada e f_{PD} é a frequência do sinal à entrada do PD.

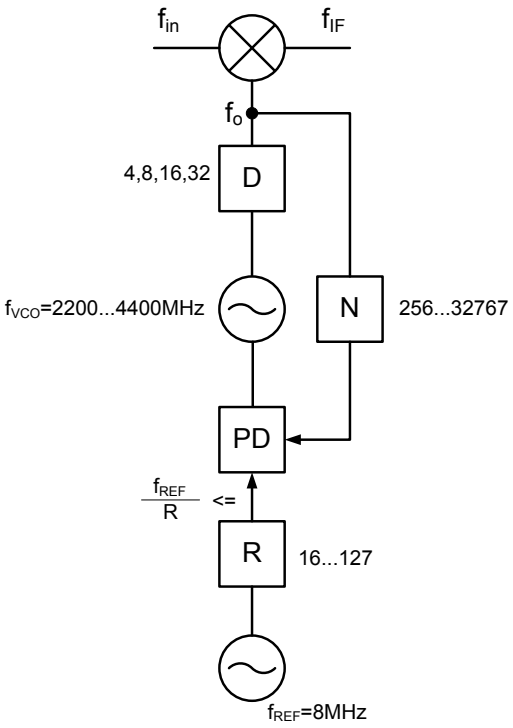


Figura 5.5: Esquema do sintetizador do kit MAX3542EVKIT

De modo a que uma frequência seja sintetizada, estas duas expressões, 5.4 e 5.5, devem ser iguais. Como o VCO apenas consegue sintetizar uma gama de frequências limitada, com a ajuda do divisor D é possível sintetizar uma gama de frequências maior. O divisor D pode tomar os valores 4, 8, 16, 32 e de modo a que a frequência desejada seja sintetizada deve ser

escolhido um D que permita que a gama do VCO fique dentro da gama em que a frequência desejada esteja, isto é, deve obedecer à seguinte equação:

$$\frac{f_{VCO_{min}}}{D} \leq f_o \leq \frac{f_{VCO_{max}}}{D} \tag{5.6}$$

A frequência à entrada do PD deve obedecer ao seguintes requisitos de frequências, para ser possível sintetizar a frequência desejada:

$$63kHz \leq f_{PD} \leq 500kHz \tag{5.7}$$

Deve-se então escolher uma frequência à entrada de PD. Como a frequência f_{PD} determina o salto possível em frequência, para uma frequência à entrada de PD escolheu-se a 100kHz. Com esta frequência é possível sintetizar frequências que distem 100kHz uma da outra. Substituindo então esta frequência nas equações 5.4 e 5.5, obtém-se então um R igual a 80 e a seguinte equação:

$$N = \frac{f_o}{100kHz} \tag{5.8}$$

O R obedece portanto às limitações do divisor que apenas pode tomar valores entre 16 e 127. O valor de N também deverá obedecer às limitações. Se não for possível sintetizar uma dada frequência deverá ser escolhido outro R de modo a que N fique dentro dos seus limites.

Um exemplo prático do que foi referido é supor que se quer sintetizar uma frequência igual a 100MHz. Como já se viu R vai ser igual a 80 enquanto que N, calculado a partir da expressão 5.8 será 1000. Será ainda preciso escolher um D para ser possível sintetizar esta frequência. Escolheu-se um D igual a 32 visto que a frequência a sintetizar fica dentro da gama que o VCO pode sintetizar.

É importante também falar de como é feita a downcovertion de RF para IF. De modo a que RF passe para uma frequência f_{IF} de 36MHz é necessário que a frequência sintetizada seja maior que a frequência de RF a transportar para IF. Esta é uma imposição feita pelo kit. Assim de modo a sintetizar a frequência correcta, a frequência do oscilador deve ser igual a:

$$f_o = f_{RF} + f_{IF} = f_{RF} + 36MHz \tag{5.9}$$

isto é, se se quiser deslocar por exemplo 100MHz para IF a frequência a sintetizar terá de ser igual a 136MHz.

Depois da frequência escolhida será necessário verificar se foi atingido o lock da PLL. Como já foi dito será preciso verificar os 3 bits do registo Status. Se for atingido o lock então foi escolhido de entre os VCOs e das suas sub bandas os correctos. Uma vez atingido o lock a frequência terá sido sintetizada com sucesso.

Falta ainda referir que este kit, tal como foi dito, possui um filtro variável em RF. Será necessário mudar este filtro sempre que se pretenda mudar de frequência RF. Os parâmetros do filtro podem ser calculados através das formulas 4.1 a 4.6 dependendo da banda de frequências a que pertence a frequência pretendida. Não será necessário ler a tabela ROM cada vez que se pretenda mudar o filtro, pois estes parâmetros guardados são constantes. É apenas necessária uma leitura de todos os registo essenciais da ROM.

Quando todas estas etapas forem cumpridas será então possível digitalizar o sinal IF. Na próxima secção explicar-se-á a maneira como as amostras vão ser retiradas.

Para simplificar o sistema foram criados buffers com os valores dos divisores R, N e D. Ainda foi criado um buffer com os parâmetros do filtro.

5.6 Controlo da amostragem

Na primeira parte deste trabalho o barramento utilizado para fazer o interface com a ADC é o protocolo SPI. O interface é muito simples baseia-se apenas num chip select e do MISO. Quando seleccionada a ADC é devolvido o valor correspondente da amostra. Como são 12 bits seriam apenas necessários 12 ciclos de relógio do barramento para enviar os dados. No entanto, a ADC envia, para além destes, 3 bits de guarda que devem ser ignorados depois de lidos, tal como se pode ver na figura 5.6.

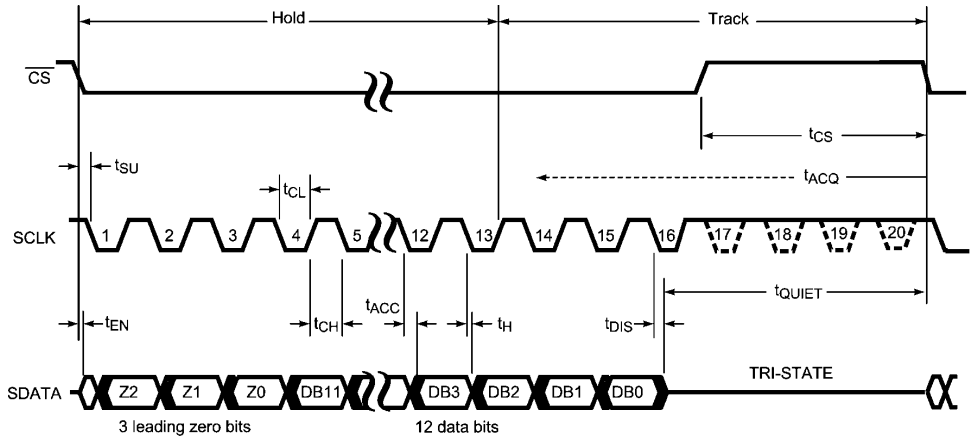


Figura 5.6: Diagrama temporal da transferência de dados da ADC121S101, retirado de [33]

A frequência com que se faz esta transferência indica a frequência de amostragem da ADC.

Na segunda parte deste trabalho vai ser utilizado o barramento FT245 Style Synchronous FIFO. As operações de escrita neste barramento são muito simples. A escrita dos dados é iniciada quando o barramento coloca TXE# a low. Através da máquina de estados é gerado o sinal WR# que inicia a escrita dos dados para o barramento. O utilizador não tem controlo nenhum sobre estas operações, pois é iniciado automaticamente pelo processador embutido no módulo. O módulo transmite logo para o PC os dados assim que recebidos. No entanto, devido à velocidade do funcionamento do USB ser inferior, o buffer interno vai encher com amostras até que fique completamente cheio.

Existe uma limitação na leitura de dados. Como apenas se conseguem enviar 510 Bytes de cada vez através do USB apenas são lidos de cada vez 510 Bytes, implicando que a amostragem feita não é contínua. Esta amostra obriga a utilizar uma FFT de 510 amostras.

Os drivers deste módulo o que faz é apenas guardar num buffer os dados que vão recebendo. Quando recebem um pedido de leitura pelo utilizador os dados que foram guardados para um buffer são lidos.

5.7 Processamento do sinal

Na segunda fase, onde é utilizado o kit MAX3542EVKIT, os dados recebidos contêm a informação do sinal IF. É necessário portanto transformá-los em informação espectral. É portanto necessário fazer uma FFT dos dados. Já foi referido na secção 3.4 como é feito o tratamento das amostra para um caso genérico. Neste caso particular a frequência de

amostragem do sinal é de 60MHz. Como IF está centrado em 36MHz e tem uma largura de banda 8MHz quando digitalizado esta banda passará para os 24MHz ocupando a gama desde 20MHz até 28MHz. Quando feita a FFT serão retirados os dados relativos a esta banda.

Como a frequência a que oscilador se encontra é superior à frequência de RF quando passa para IF o espectro aparece invertido. Ao ser digitalizado ele é invertido novamente, retornando assim à sua posição original.

Na próxima secção falar-se-á do algoritmo geral do controlo.

5.8 Algoritmo de controlo

Relativamente à primeira fase desta dissertação, onde se vai utilizar um front-end concebido no âmbito de outra dissertação de mestrado[2], o controlo efectuado ao sistema é apresentado no diagrama de blocos da figura 5.7.

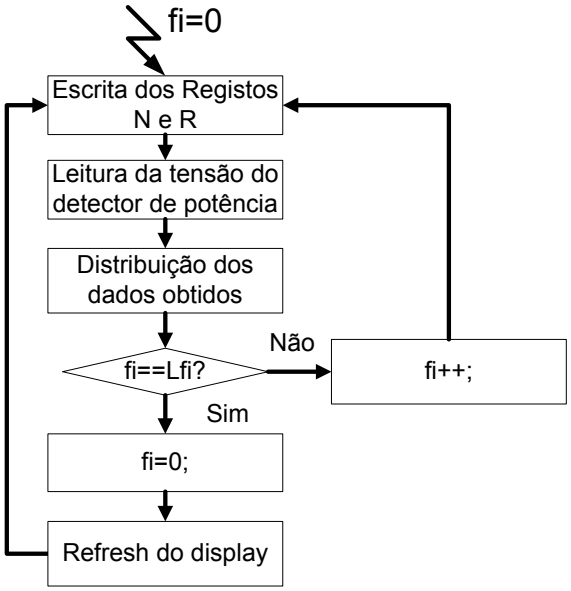


Figura 5.7: Diagrama de blocos do controlo

Em primeiro lugar é de referir que os valores que iram tomar os divisores R e N serão guardados em buffers. Cada posição deste buffer pertence a uma frequência a sintetizar e os valores estão ordenados por grandeza da frequência, da mais baixa para a mais alta. O índice fi aponta para a frequência a sintetizar e quando iniciado o controlo é colocado o índice a zero de modo a que se comece na menor frequência. Depois de iniciado o controlo, primeiro são escritos nos registos N e R seguido da leitura do sinal proveniente do detector de potência. Depois de feita a média aos dados obtidos, esta será colocada no buffer de display na posição relativa à frequência sintetizada. Chegado a este ponto se a última frequência foi atingida será necessário colocar o índice fi na posição inicial e fazer o refresh ao display, caso contrário será incrementado o fi que corresponde ao aumentar da frequência.

É agora apresentada na figura 5.8 o diagrama de blocos do controlo efectuado na segunda fase, onde é utilizado o kit MAX3542EVKIT. O sistema é iniciado colocando o fi a zero. Este fi é índice dos buffers que contêm os divisores R, N e D e ainda dos buffers que contêm

os parâmetros para o filtro. Considerando que os divisores e os parâmetros foram colocados por ordem crescente, colocando o índice fi a zero está-se a sintetizar a primeira frequência ou seja neste caso a colocar a primeira banda de frequências a IF.

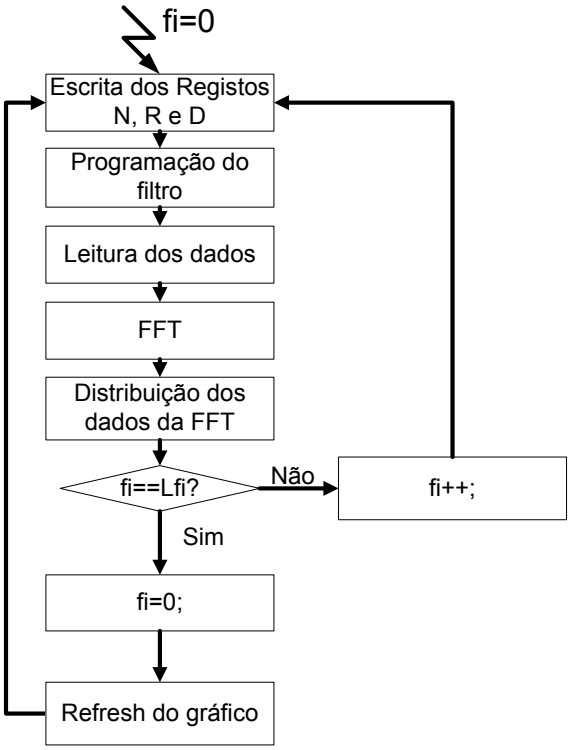


Figura 5.8: Diagrama de blocos do controlo efectuado

Depois são escritos os registos R, N e D nos registos do kit. Para além destes registos também são escolhidos o VCO e a sua sub banda. De seguida são escritos os parâmetros do filtro adaptando assim o sistema a esta banda de frequências. Depois de configurados estes parâmetros, pode ser feita a leitura do sinal IF. São de seguida tratados os dados fazendo a FFT e colocando no sítio correcto do buffer do eixo dos yy os dados relativos às frequências provenientes da FFT tendo em conta a banda de frequências que está a ser digitalizada.

Chegado a este ponto é necessário verificar se se chegou à última banda a digitalizar. Se tal acontecer o índice deve ser colocado a zero, voltando à banda inicial e de seguida, feito o refresh do Display. Caso contrário é incrementado o índice, passando a banda a digitalizar para uma banda superior. No final o sistema volta ao início do diagrama de blocos.

Capítulo 6

Resultados

Neste capítulo são apresentados os resultados do funcionamento do analisador de espectros. A primeira fase deste trabalho não pode ser concluída, devido ao facto que um dos componentes constituintes do front-end, mais concretamente o sintetizador, não estar a funcionar correctamente. Uma vez que este não pode ser controlado não foi possível pôr em funcionamento o sistema em total. No entanto foi possível verificar o funcionamento da placa que inclui a ADC ADC121S101. O seu funcionamento será falado na seguinte secção.

Na segunda fase do trabalho verificou-se o funcionamento correcto do analisador. Primeiro verificou-se o correcto funcionamento da placa desenvolvida, isto é, o funcionamento da máquina de estados e da ADC ADC08100. Por fim foi verificado o funcionamento do sistema total.

Para os testes desta segunda parte foram utilizados os seguintes instrumentos:

- Logic Analyser - Agilent 16822A
- Gerador de sinais - Wiltron Model 6769B

6.1 Placa com a ADC121S101

Tal como já foi referido apesar de não ser possível implementar o sistema do analisador de espectros, testou-se apenas o funcionamento da placa criada para conter à ADC ADC121S101.

Um teste simples consiste em aplicar uma tensão constante tal como aquela que seria aplicada à entrada da ADC. Recorreu-se então a um voltímetro e criou-se um programa que apenas acede-se a ADC. A partir deste programa, no qual é feita a conversão dos dados da ADC para a tensão correspondente, foi feita uma recolha de tensões medias lidas pela ADC e comparadas com as tensões lidas de através de um voltímetro. Este resultado é apresentado na figura 6.1. Foi feita uma análise do seu erro absoluto e do seu erro relativo respectivamente figura 6.2 e 6.3. Pode-se concluir que os resultados apresentados são bastante satisfatórios visto que o erro relativo não vai além de os 0,35%. Pode-se criar então um analisador bastante preciso desde que o detector de potência seja linear dentro desta gama de voltagens.

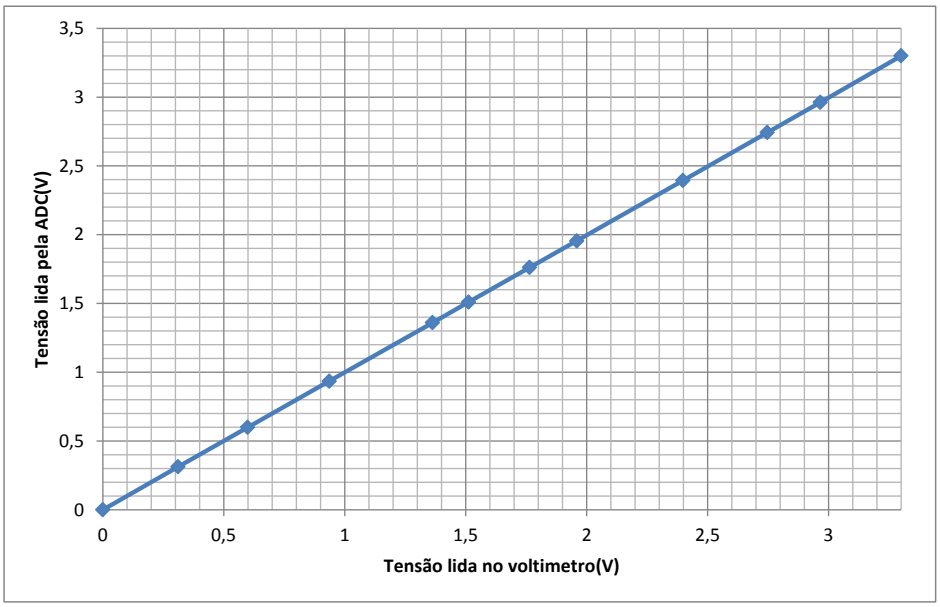


Figura 6.1: Tensão lida no Voltímetro vs ADC

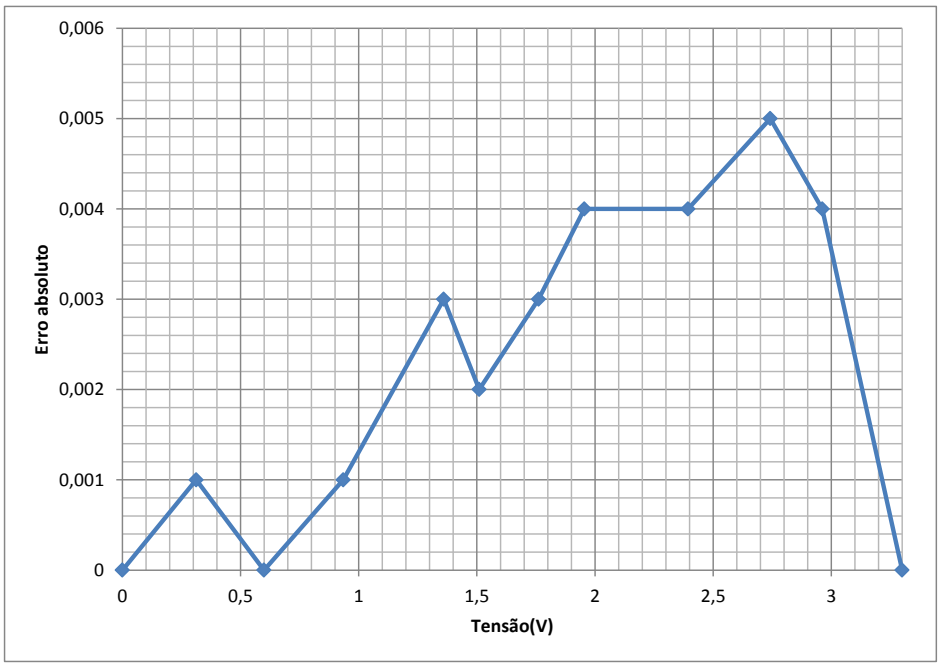


Figura 6.2: Erro absoluto

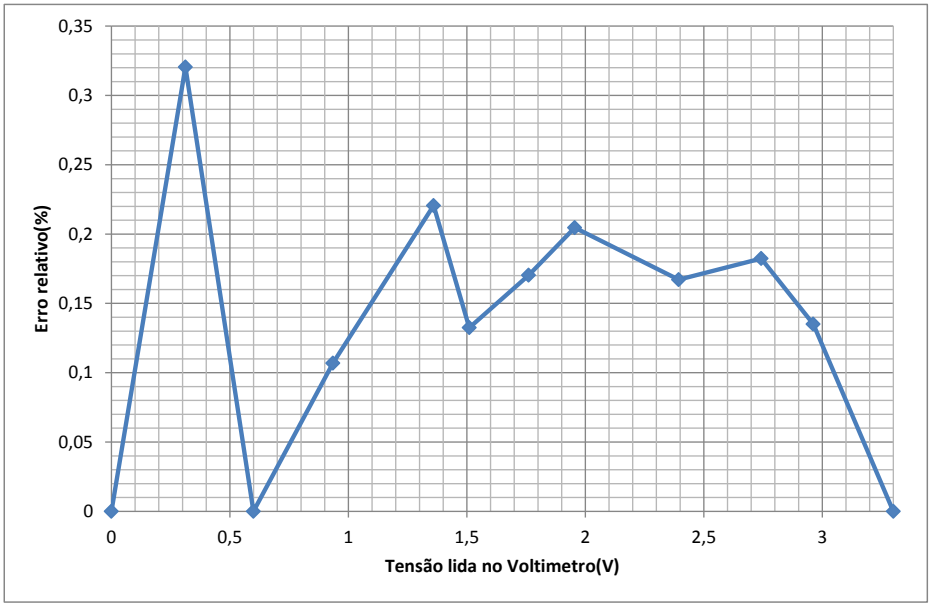


Figura 6.3: Erro relativo

6.2 Placa com a ADC08100

6.2.1 Maquina de estados

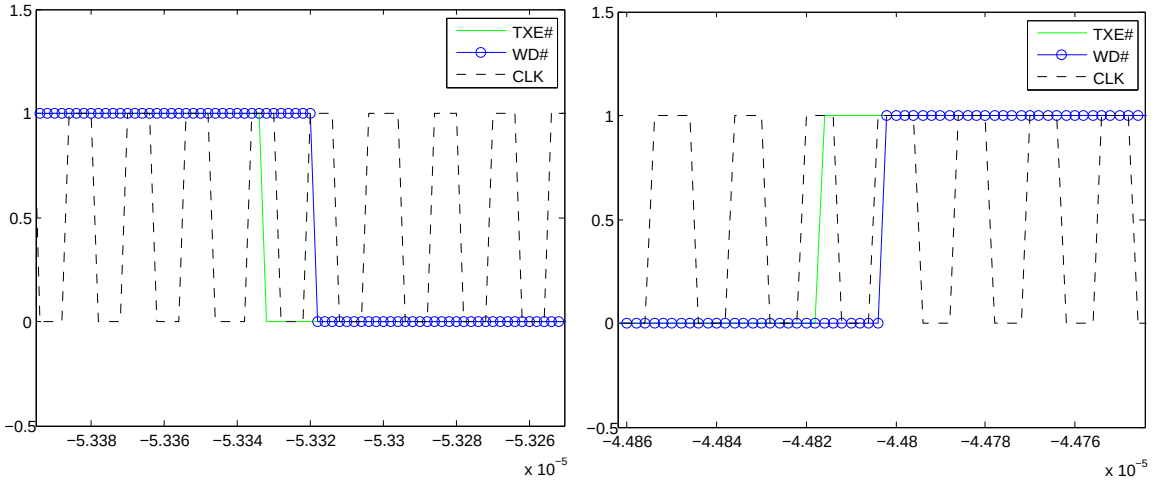
Nas figuras 6.4 é apresentado o diagrama temporal da máquina de estados implementada. Os dados provêm do Logic Analyser Agilent 16822A e foi feita uma amostragem ao sinal a 2ns. Comparando estes diagramas com o diagrama projectado (figura 4.14) pode-se ver que são ambos muito semelhantes. No entanto, é necessário garantir que o sinal vá a Low pelo menos 11ns antes do próxima transição de Low para High. Através da ajuda do Matlab© foi possível determinar se eram atingidos estes requisitos. O máximo de tempo atingido desde que se deu a transição de Low para High do clock até que o sinal WD# transita-se de High para Low foi 2ns, que correspondente ao tempo de amostragem. Este valor pode não ser o correcto uma vez que o tempo pode ser menor.

Se se sabe que um ciclo de relógio tem um período de 16.667ns, conclui-se que o tempo mínimo para que o sinal WD# vá a Low antes da próxima transição de clock é de 14.667 que é maior que os 11ns permitidos. Foi portanto implementado com sucesso a máquina de estados da placa.

6.2.2 ADC

Uma vez verificado o funcionamento da máquina de estados, é necessário verificar o funcionamento correcto da ADC. Uma vez que não é possível aplicar um sinal DC à entrada da placa, pois foi concebida de modo a que a componente DC do sinal à entrada seja filtrada, foi aplicado um sinal sinusoidal de modo a testar o seu funcionamento.

Na figura 6.5 estão expostos os dados sem qualquer tipo de tratamento recebidos através do USB, provenientes da ADC. Os dados são referentes a um sinal sinusoidal de 500kHz com uma potência de -3dBm. Como se pode ver o sinal tem a forma de uma sinusóide o



(a) Transição de High para Low

(b) Transição de Low para High

Figura 6.4: Diagrama temporal da maquina de estados implementada

que permite constatar que a ADC esta a amostrar correctamente o sinal. No entanto, é de reparar que existe uma descontinuidade da amostra 510 para a 511 como se pode ver na figura 6.6. Esta pequena descontinuidade não se trata de erro na amostragem mas sim devido a que o barramento que lê da ADC apenas conseguir ler de cada vez 510 Bytes. Tal pode ser verificado na transição da amostra 1020 para a 1021. Devido a este pormenor, a precisão do espectro será limitada a $\frac{60MHz}{510} = 117,6kHz$.

6.3 Analisador de espectros

Nesta secção serão apresentados todos os resultados referentes ao funcionamento do analisador de espectros projectado na segunda parte desta dissertação.

Antes de efectuar-se as medições, calibrou-se o sistema de modo a que a potencia aplicada a entrada corresponde-se ao apresentado no analisador de espectros. O resultado desta calibração é apresentado nas figuras 6.7 a 6.10 no qual foi aplicado um sinal sinusoidal com uma frequência de 101MHz à entrada do sistema. Como se pode ver os resultados da calibração para esta frequência são bastante bons uma vez que a potencia do sinal aplicado à entrada corresponde ao apresentado no interface gráfico.

No entanto devido a problemas de saturação e de precisão a escala apenas é constante para esta frequência desde -40dBm a -75dBm. Acima de -40dBm o sistema satura, como se pode ver nas figuras 6.11 e 6.12 em que respectivamente são aplicados sinais com uma potência de -20dBm e -30dBm. Esta saturação deve-se ao facto de se estar a utilizar o Automatic Gain Control (AGC) em conjunto com o detector de potência do kit Max3542EVKIT. Quando é detectado um certo nível de potência, o comparador que está ligado a tensão que controla o AGC começa a diminuir a tensão aplicada diminuindo o seu ganho deste modo quando é aplicado um sinal de -20dBm à entrada do kit o ganho do AGC diminui de modo a que a saída do kit estejam aproximadamente 0dbm.

Uma vez que o ganho aumenta ou diminui consoante o sinal aplicado para a banda IF, o sinal não é o único a ser amplificado, o ruído é também amplificado. Este efeito pode ser

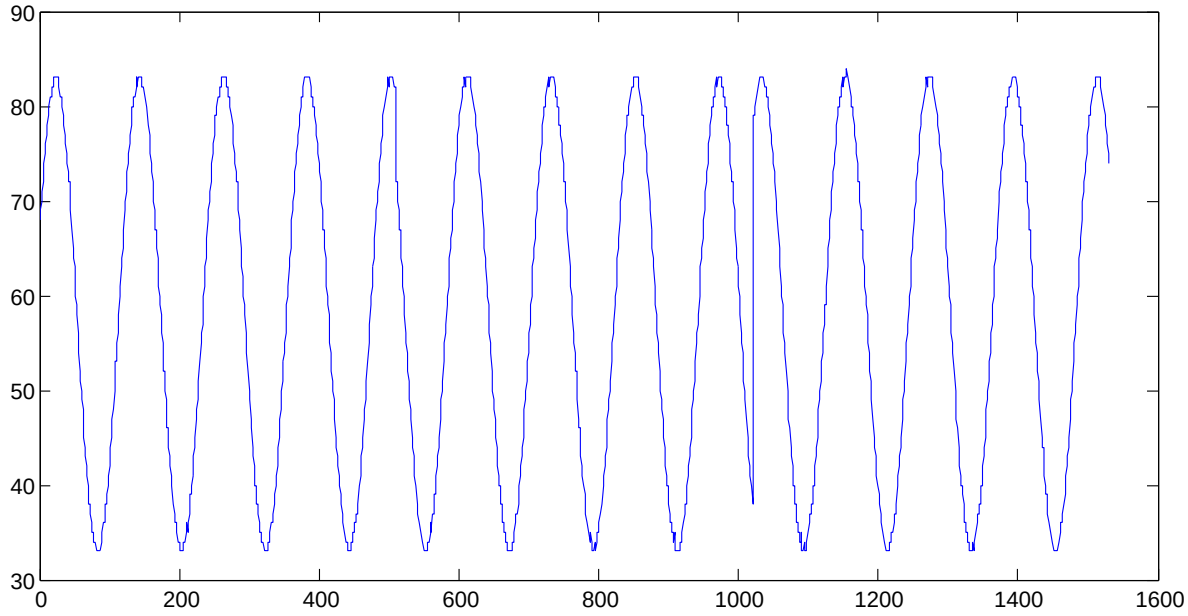


Figura 6.5: Sinal sinusoidal de 500kHz com uma potência de -3dBm aplicado a entrada da placa

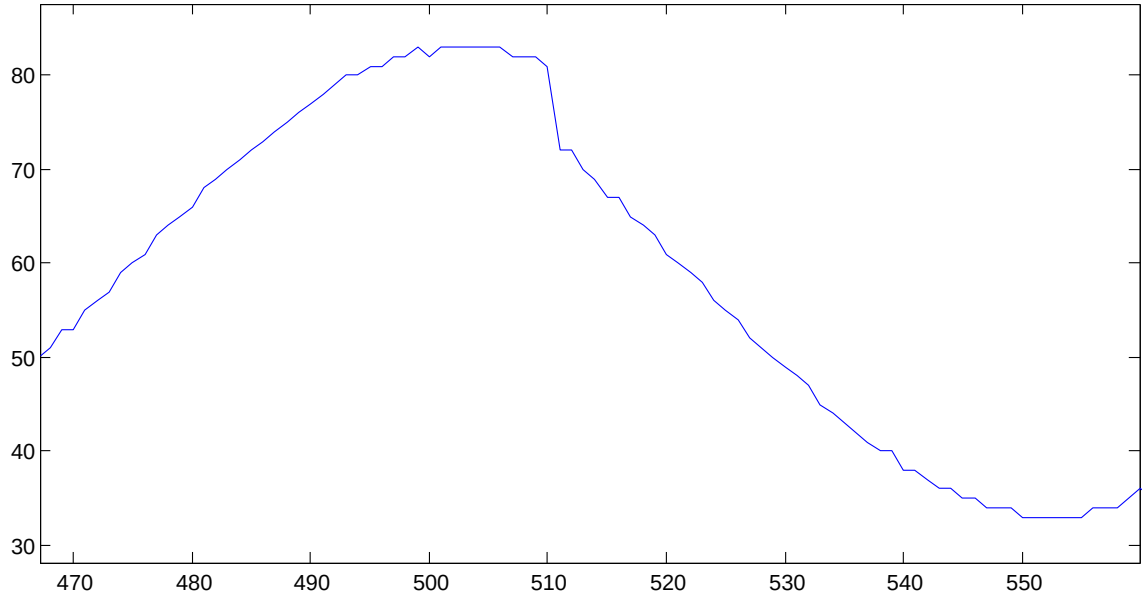


Figura 6.6: Sinal sinusoidal de 500kHz com uma potência de -3dBm aplicado a entrada da placa com mais pormenor

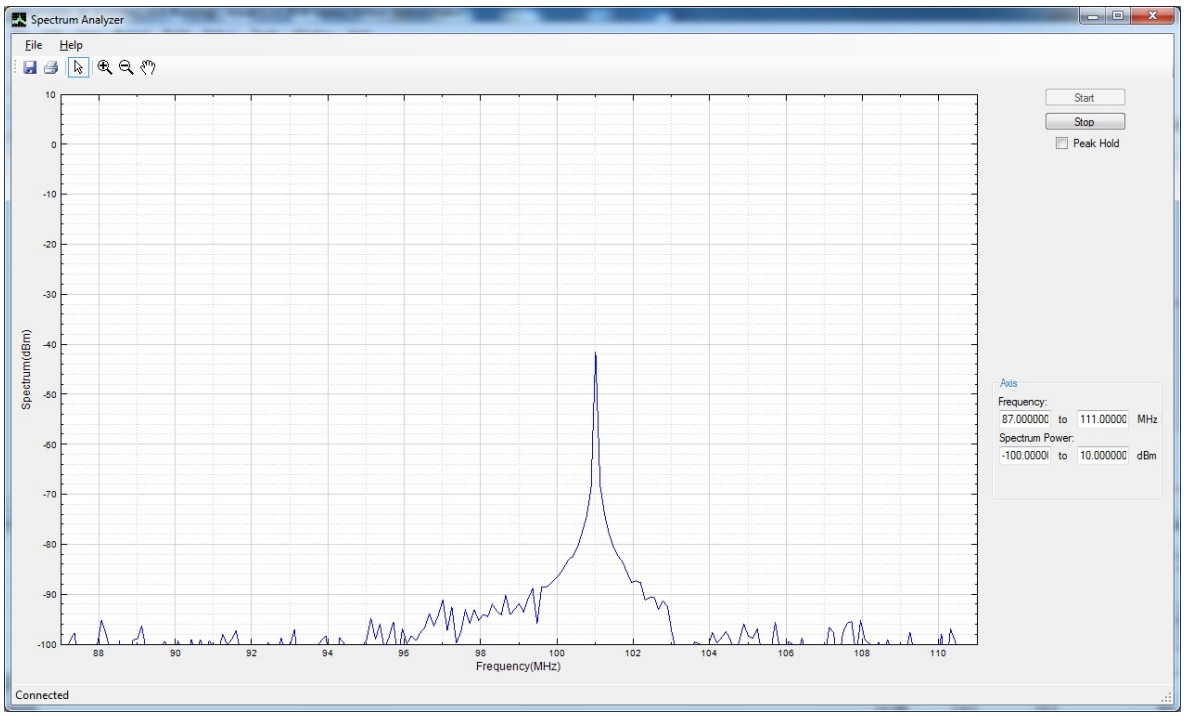


Figura 6.7: Analisador de espectros com um sinal de entrada de 101MHz e -40dBm

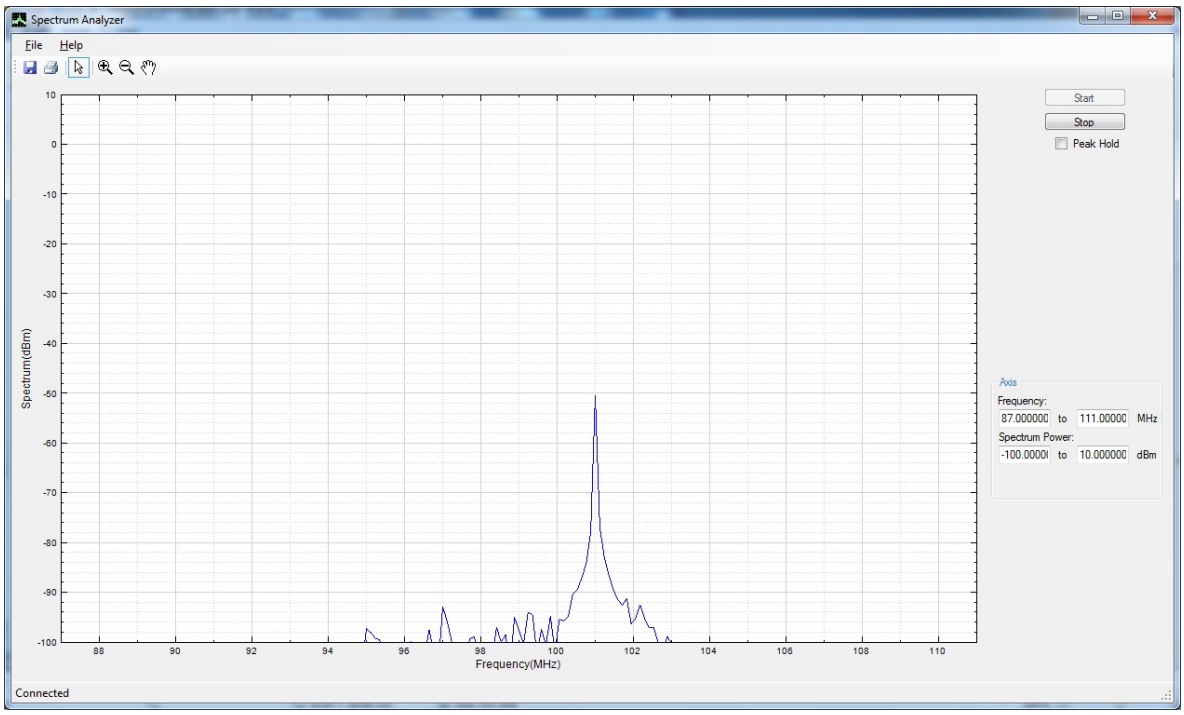


Figura 6.8: Analisador de espectros com um sinal de entrada de 101MHz e -50dBm

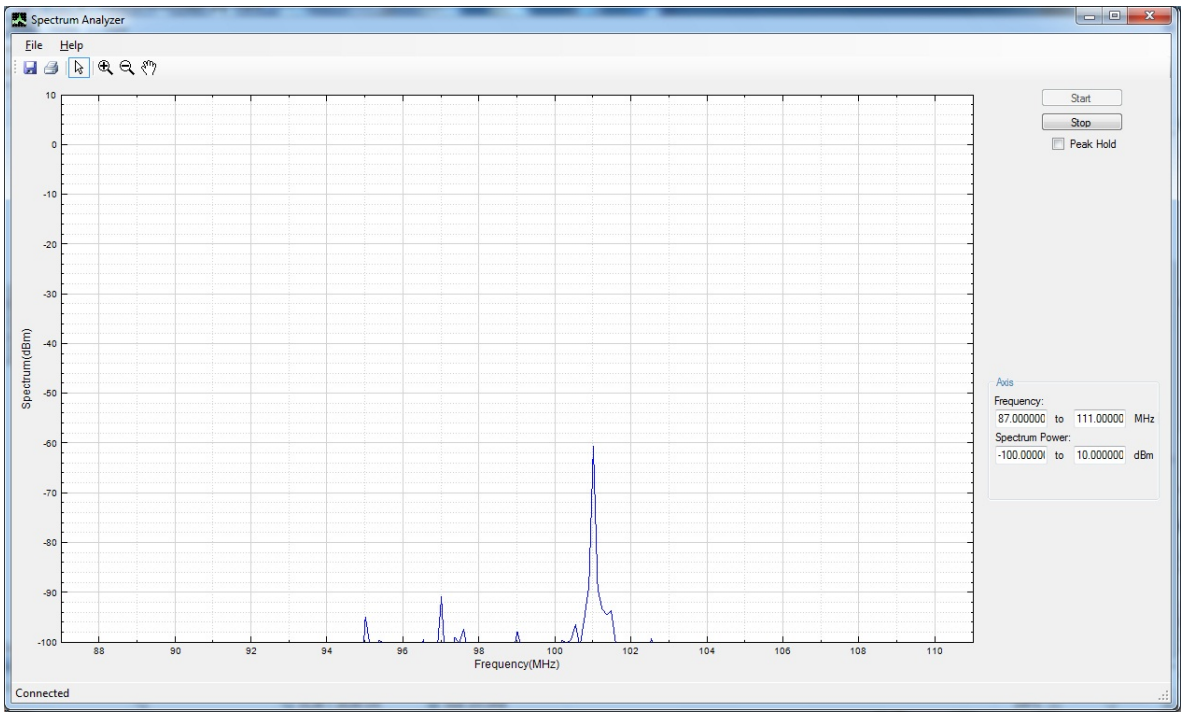


Figura 6.9: Analisador de espectros com um sinal de entrada de 101MHz e -60dBm

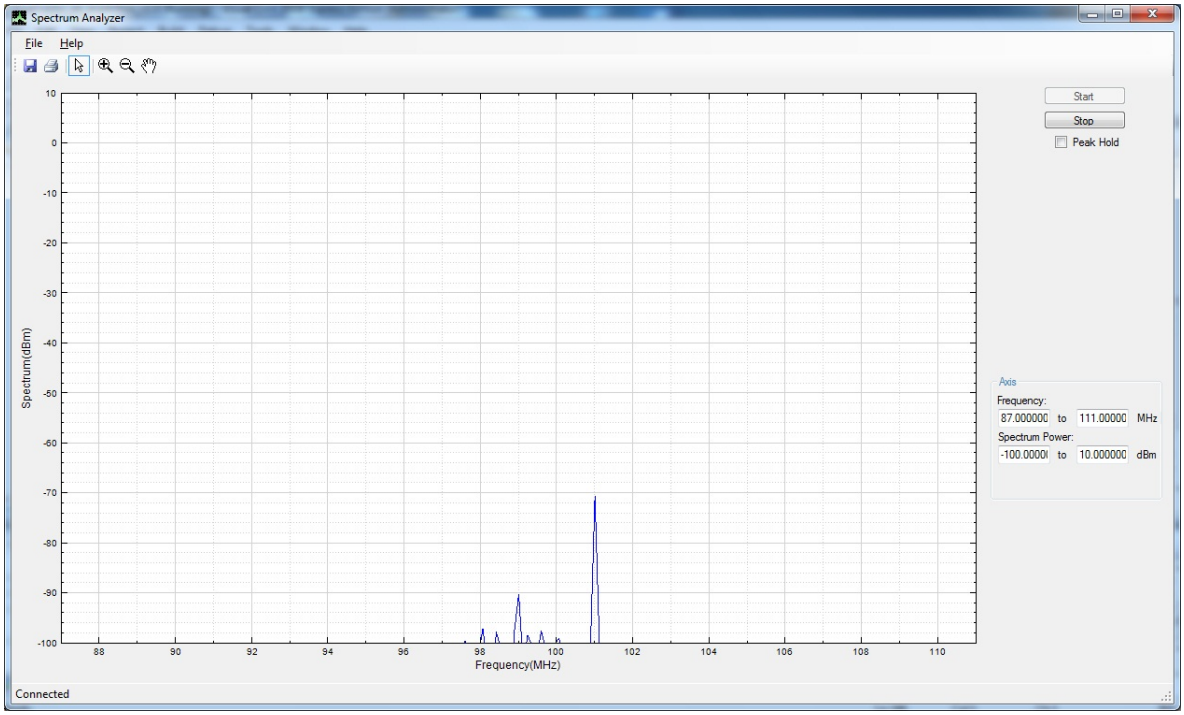


Figura 6.10: Analisador de espectros com um sinal de entrada de 101MHz e -70dBm

observado em todas as figuras 6.7 a 6.10 mas mais concretamente nas figuras 6.7 e 6.8 em que este efeito pode ser observado.

Devido a problemas de precisão da ADC abaixo de -70dBm o sistema não funciona correctamente tal como se pode ver nas figuras 6.13 e 6.14.

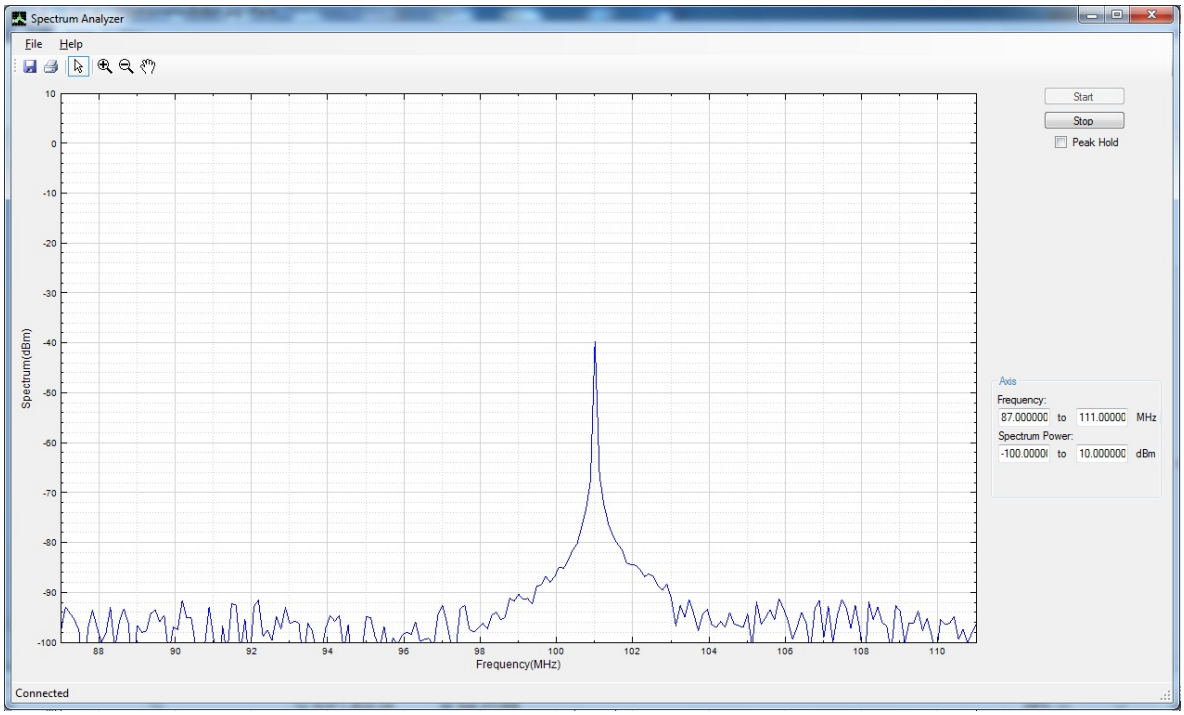


Figura 6.11: Analisador de espectros com um sinal de entrada de 101MHz e -20dBm

No entanto um analisador de espectros não analisa apenas uma frequência, mas sim uma gama de frequências. Para verificar o comportamento do sistema aplicaram-se 3 sinais sinusoidais com distintas frequências. Primeiro foi aplicado um sinal com uma frequência de 91MHz e uma potencia de -50dBm tal como se pode ver na figura 6.15. Como se pode ver pela figura o sistema continuou a responder correctamente ao sinal de entrada. Decidiu-se então aplicar um sinal com uma frequência de 92MHz mantendo a mesma potência de -50dBm. O resultado apresentado pelo sistema foi o da figura 6.16 e como se pode ver neste caso o resultado não corresponde a potência do sinal aplicado uma vez que a precisão do sistema em termos de frequência é bastante baixa. A frequência de 92MHz esta entre dois pontos das frequências que se podem analisar não sendo possível obter uma leitura correcta do valor de potência para esta frequência.

Para o terceiro sinal foi aplicado um sinal de 95Mhz com uma potência de -50dBm o resultado obtido foi o apresentado na figura 6.17. Como se pode ver também neste caso o sinal não corresponde ao esperado. Neste caso a causa para o sinal ter uma potência mais baixa, deve-se ao facto de se estar a trabalhar numa das zonas periféricas ao final da banda de passagem do filtro aplicado ao sinal IF, nas quais é mais acentuada as atenuações ao sinal.

Para melhor compreender o sistema, tentou-se perceber o seu comportamento consoante a frequência do sinal aplicado, mantendo sempre a potência constante. Decidiu-se usar um sinal com uma potência de -50dBm, e utilizar um passo de 1MHz. O resultado deste estudo foi o apresentado na figura 6.18. Pela figura pode-se concluir que tal como foi dito anteriormente

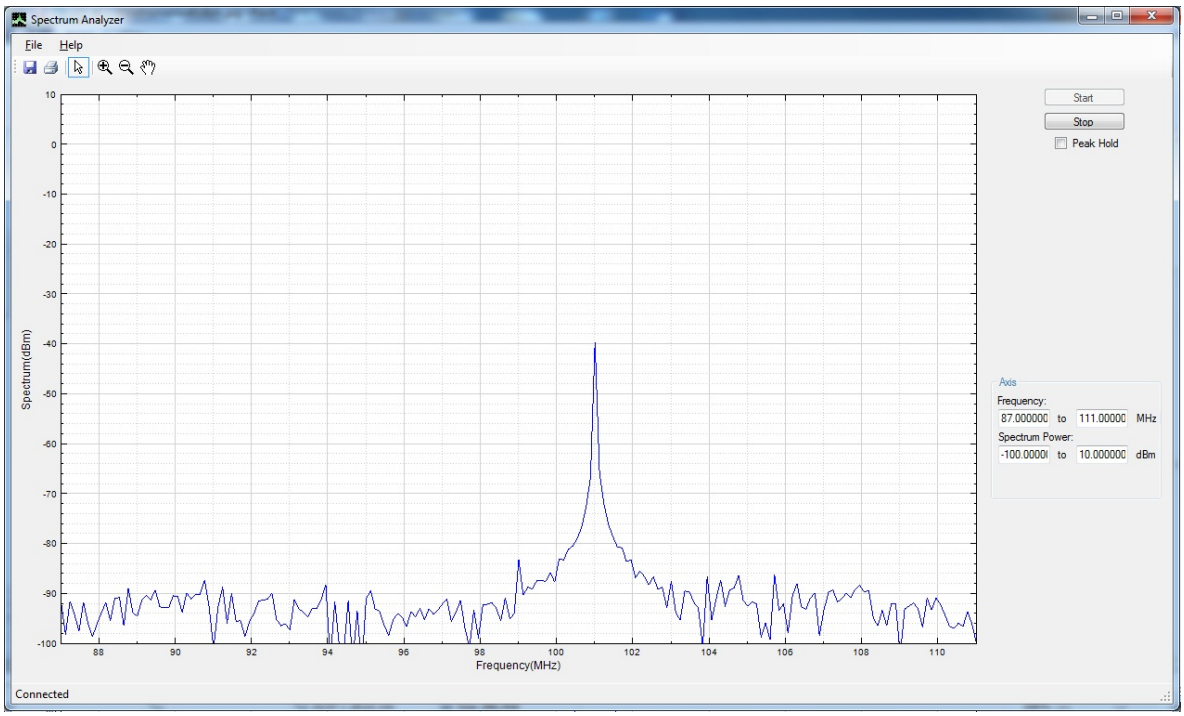


Figura 6.12: Analisador de espectros com um sinal de entrada de 101MHz e -30dBm

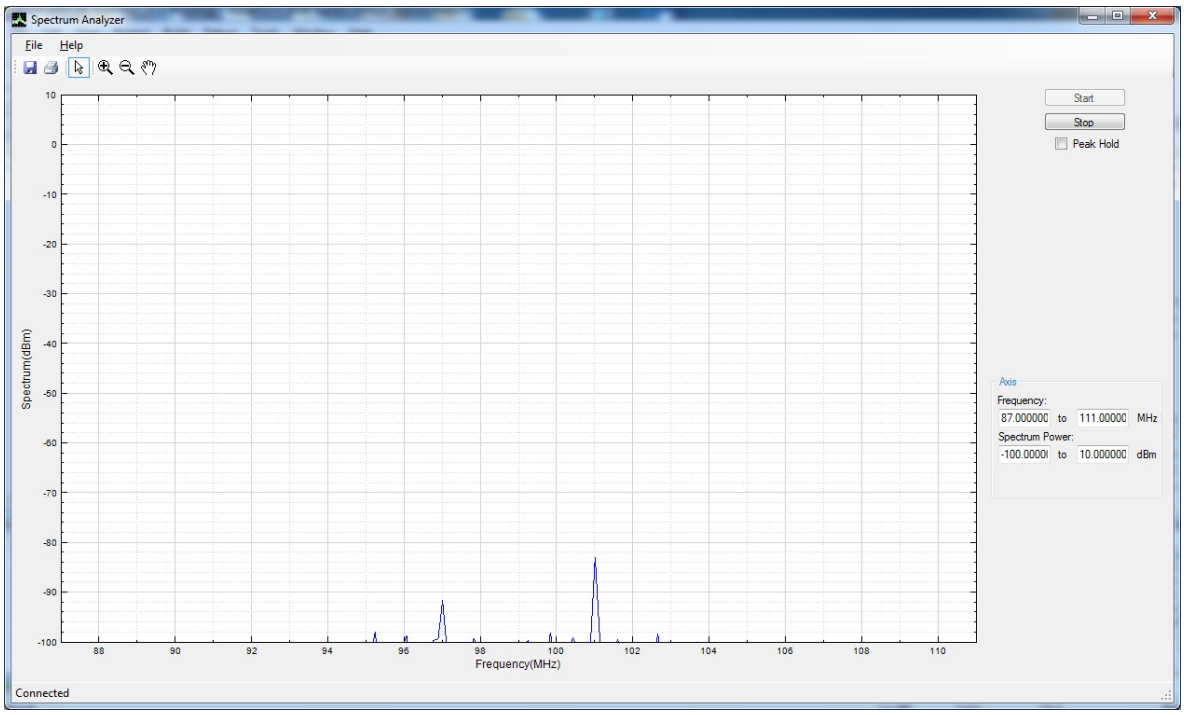


Figura 6.13: Analisador de espectros com um sinal de entrada de 101MHz e -80dBm

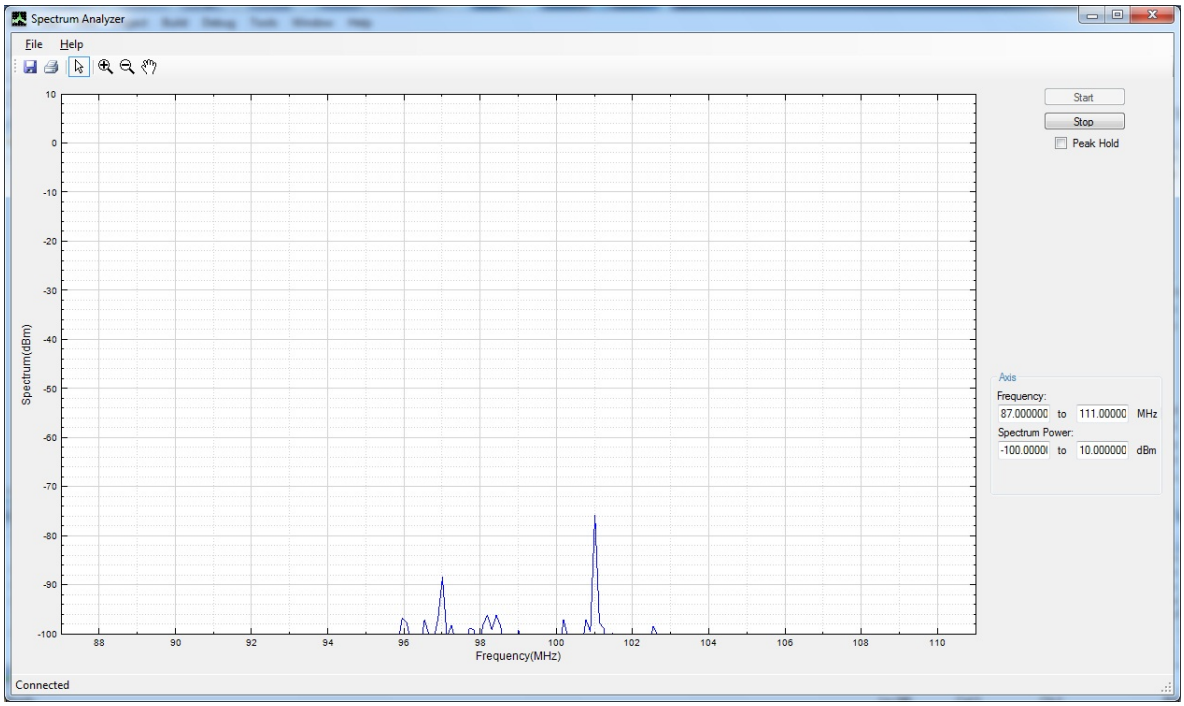


Figura 6.14: Analisador de espectros com um sinal de entrada de 101MHz e -75dBm

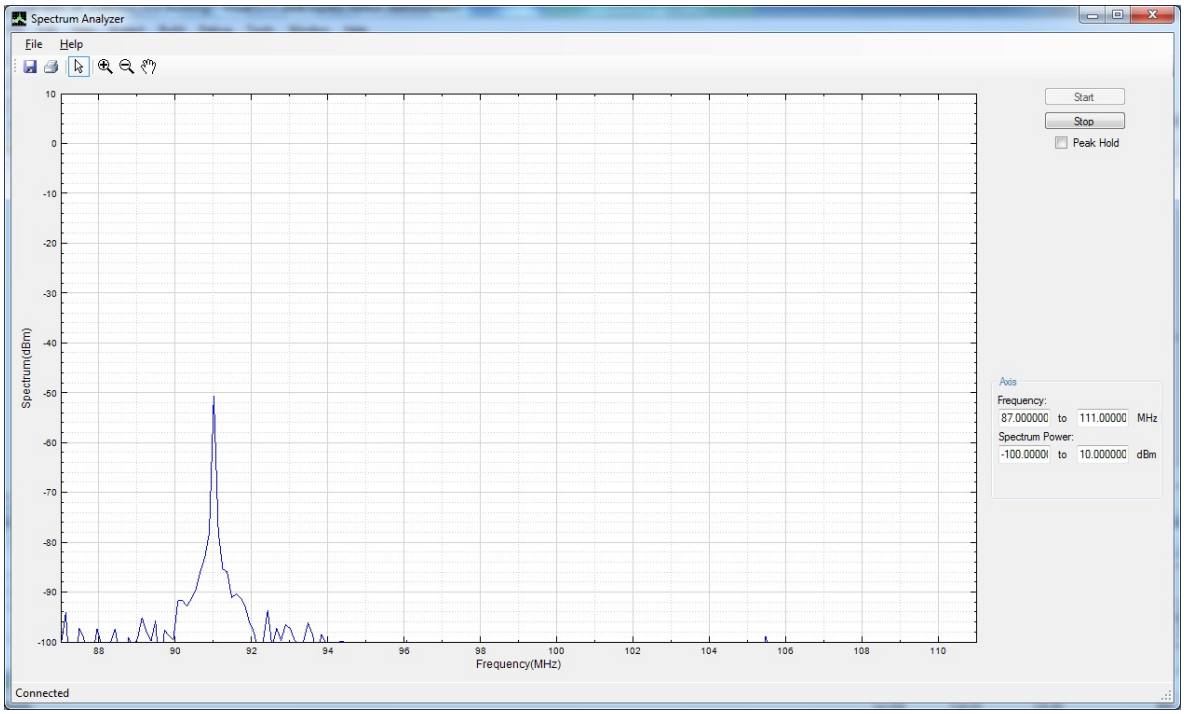


Figura 6.15: Analisador de espectros com um sinal de entrada de 91MHz e -50dBm

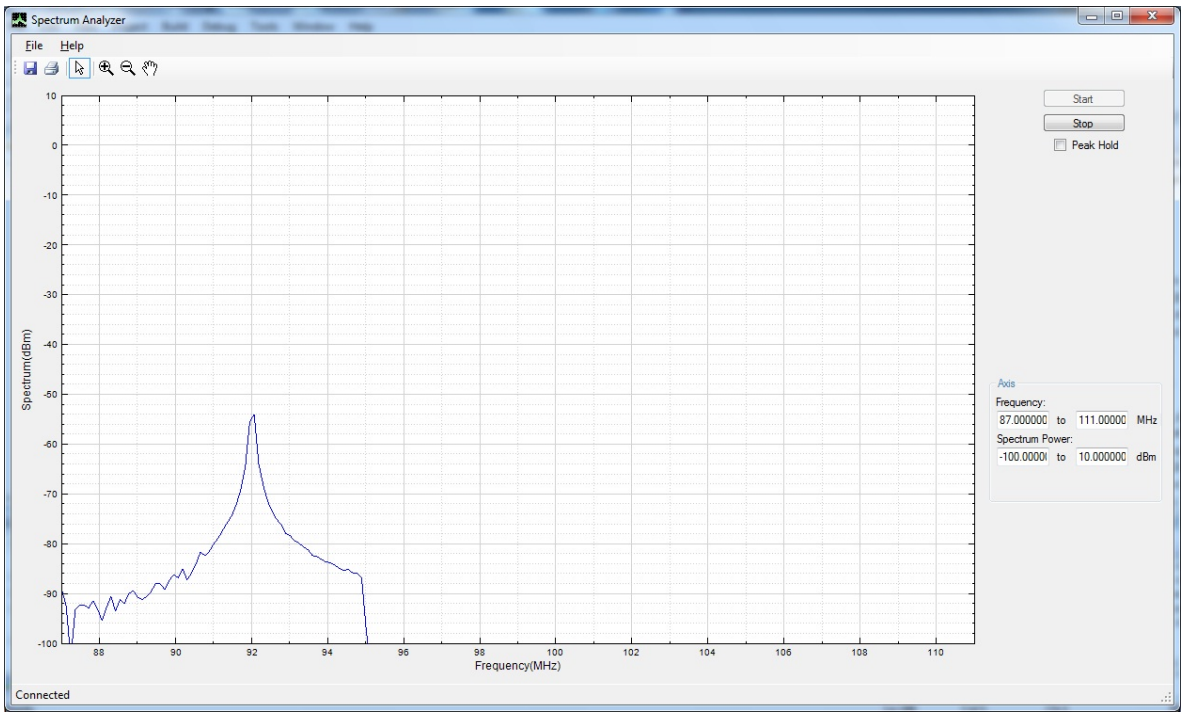


Figura 6.16: Analisador de espectros com um sinal de entrada de 92MHz e -50dBm

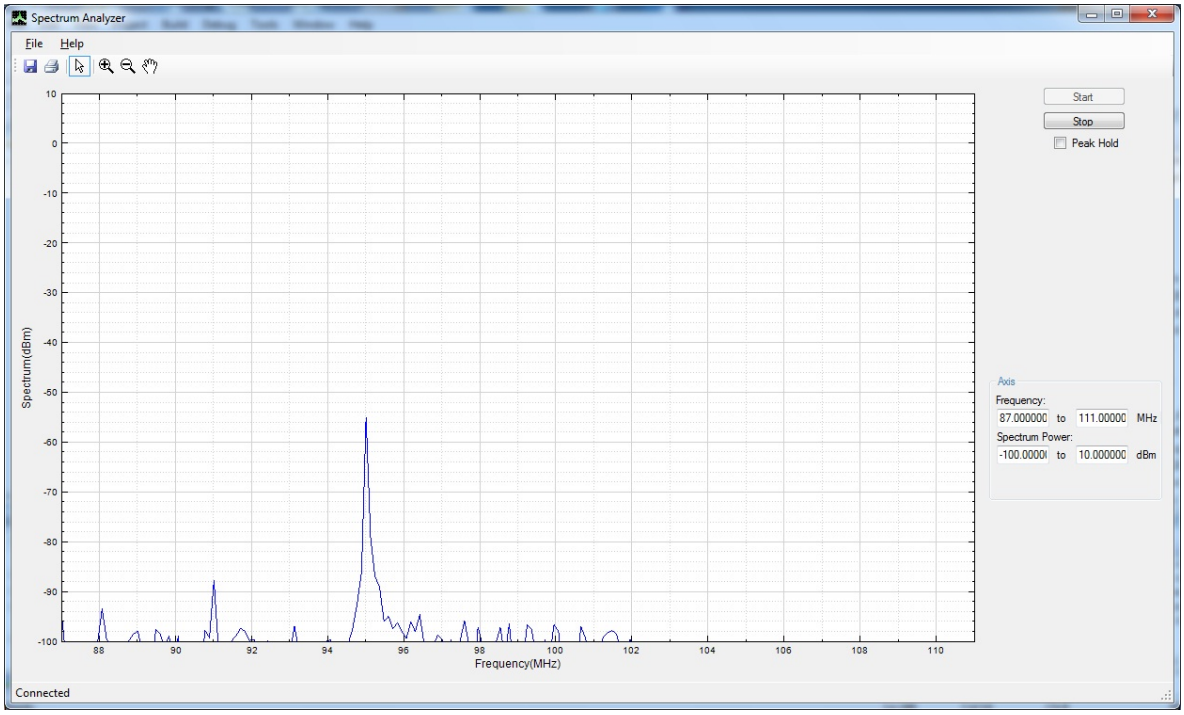


Figura 6.17: Analisador de espectros com um sinal de entrada de 95MHz e -50dBm

o sistema junto do final da banda de passagem atenua o sinal, isto é em 87MHz, 95MHz, 103MHz e 111MHz. Também é de referir que em todas as frequências pares o sinal não consegue ser amostrado correctamente devido a falta de precisão do sistema. Nas frequências ímpares a excepção das referidas o sinal é amostrado muito próximo do real, apenas havendo uma discrepância máxima de 1dBm em torno do valor ideal.

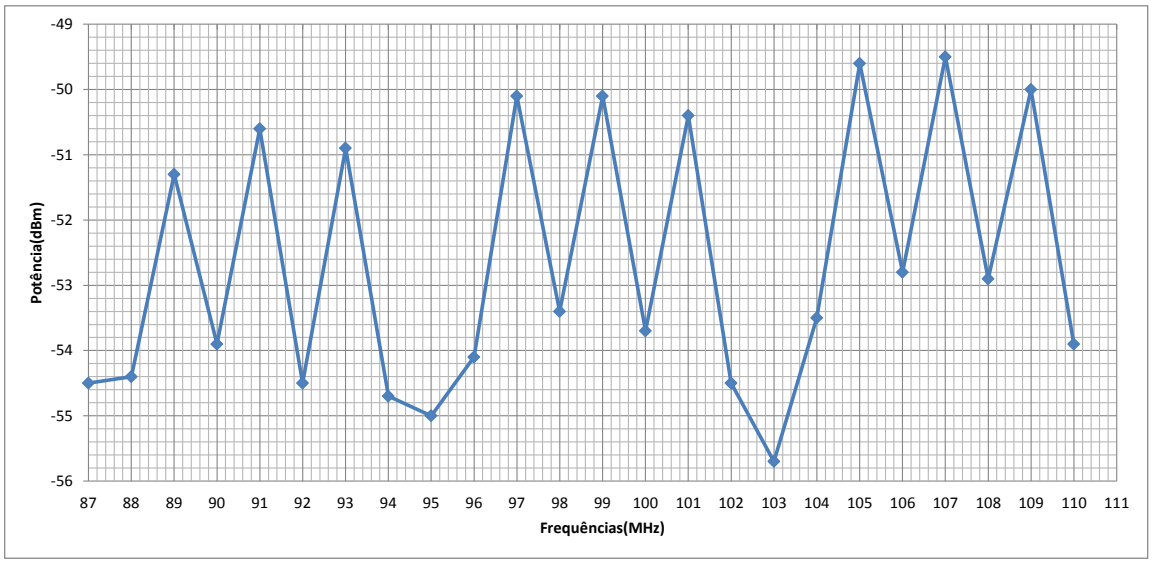


Figura 6.18: Comportamento do sistema para as várias frequências mantendo a potência do sinal constante

O problema da banda de passagem pode ser resolvido através da alteração das bandas a passar para IF e do espectro retirado da FFT dos dados provenientes da ADC, isto é, em vez de ser retirado os 8MHz apenas serão retirados 6MHz eliminando assim os dados que ficariam no fim de banda de passagem do filtro aplicado ao sinal IF.

Na figura 6.19 pode-se observar que foi minimizado o problema. No entanto o melhoramento apenas foi de 2dB.

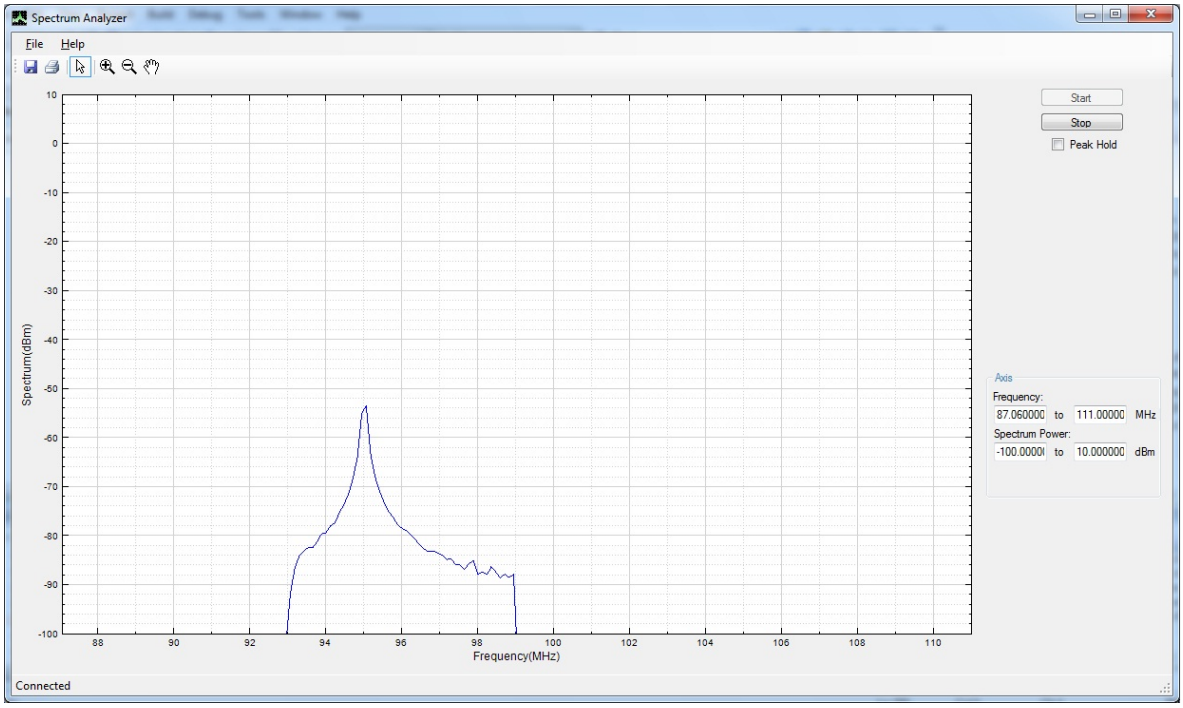


Figura 6.19: Analisador de espectros com um sinal de entrada de 95MHz e -50dBm

Capítulo 7

Conclusões

O trabalho desenvolvido nesta dissertação, tinha como objectivo criar dois analisadores de espectros para gamas diferentes. No entanto para a banda de 2.125GHz até 2.6GHz, o analisador não podê ser finalizado, devido a problemas com o sintetizador da parte do front-end. Apesar disto foi desenvolvido com sucesso um pequeno módulo que permite ler uma tensão constante, utilizando uma ADC com 12 bits a ADC121S101.

Para a segunda fase do trabalho, foi possível implementar um analisador de espectros baseado em SDR. O interface gráfico criado, apesar de alguns bugs, funciona dentro do esperado. Não foi possível, no entanto, ter uma grande precisão em termos de frequências, e a gama de potências a que opera correctamente é de apenas -70dBm até -40dBm. Ainda assim o sistema só é preciso nas frequências em que o sinal é múltiplo de 117,6kHz. Apesar destas limitações o sistema foi implementado com sucesso, constituindo um progresso no desenvolvimento deste tipo de sistemas. Foi provado ainda que é possível implementar o sistema só que com reduzida precisão e para uma reduzida gama de potências.

7.1 Trabalho Futuro

Para um trabalho futuro seria bom implementar entre o USB e a ADC uma Field-programmable gate array (FPGA). Esta FPGA será encarregue de tratar parte do sinal IF digitalizado. Deverá fazer a análise espectral do sinal recebido através da FFT. De modo a que seja implementada a FFT é necessário estudar a maneira mais viável e que utilize o menor número de recursos, no entanto deverá ser mais precisa que o sistema implementado. Deverá utilizar-se uma placa que já tenha implementado um controlador USB. Para este caso poderão ser utilizados módulos da KNJN como por exemplo a Saxo-Q FPGA em que já estão incluídas 4 ADCs de 8bits e uma frequência de amostragem de 200MSPS e 2 DAC de 10 bits com uma frequência de amostragem de 165Mega Samples Per Second (MSPS) [34]. Ainda existem placas deste produtor que são bastante interessantes, no entanto, seria necessário implementar a ligação com as ADCs. Para além da KNJN existem outras placas bastante interessantes apresentadas pela Digilent, no entanto, sofrem o mesmo problema de ser necessário implementar uma ligação com a ADC.

Poderá ser realizada ainda uma amostragem de bandas diferentes com ADCs, isto é, receber duas bandas de frequências distintas de modo a que não haja sobreposição de bandas e que aumente a banda de observação. Isto pode ser muito benéfico pois é reduzido o tempo de análise do espectro por um factor de dois, isto se for possível paralelizar estas duas amos-

tragens. No entanto a quantidade de dados provenientes da FPGA seria um factor limitador deste sistema, pois a velocidade a que se consegue comunicar com o PC tinha de ser aumentada.

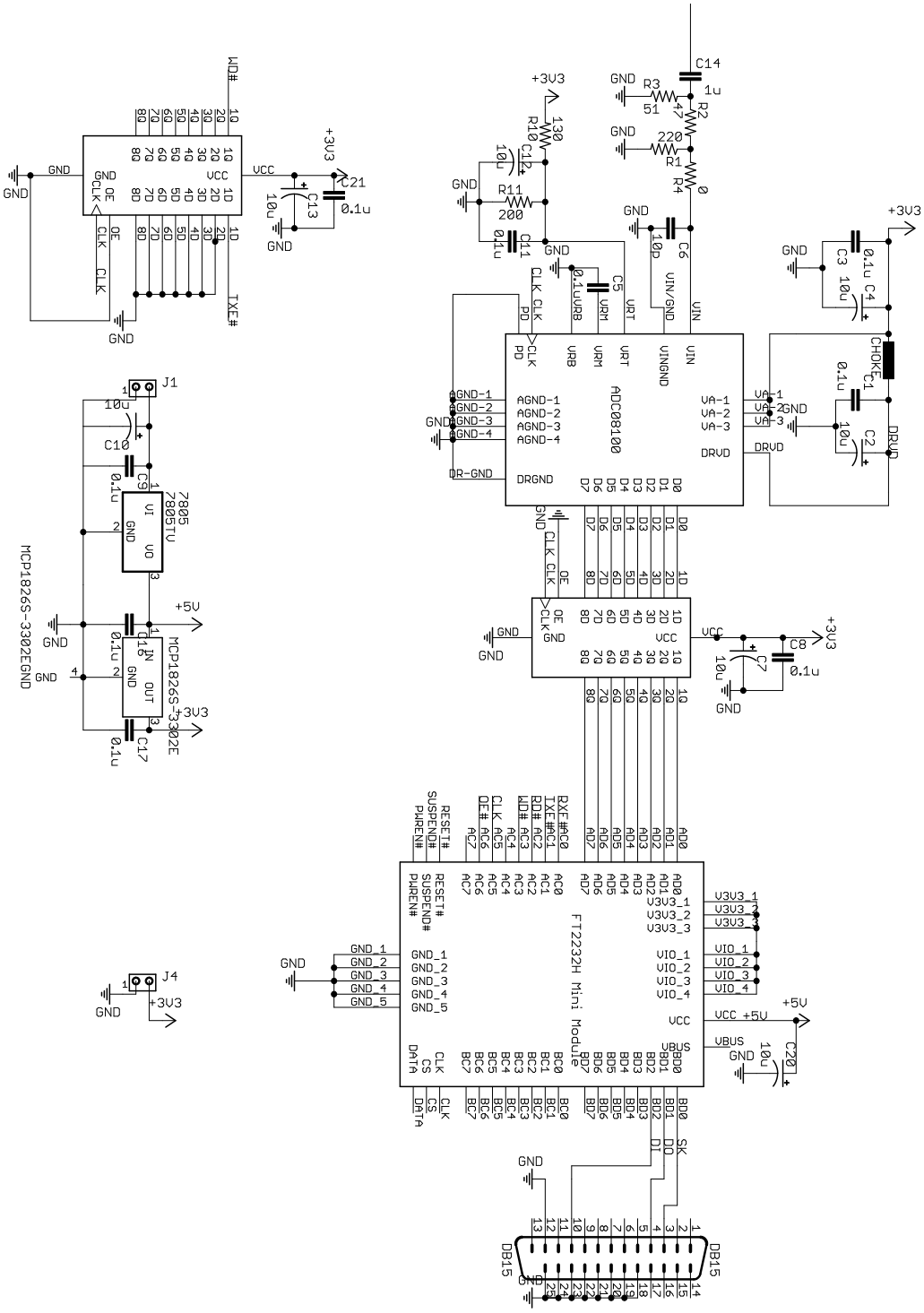
Para que a velocidade de comunicação aumentasse seria necessário que o barramento escolhido conseguisse suportar a enorme quantidade de informação proveniente não só da FPGA, mas também do PC no caso de ser necessário controlar um ou dois osciladores. A necessidade de utilizar um ou dois osciladores dependeria não só da banda a analisar mas também da frequência de amostragem da ADC e da largura de banda dos componentes críticos do sistema para transportar a banda para IF. Uma das hipóteses seria a evolução do USB 2.0 para 3.0. Este barramento, para além de suportar transferência de dados 10 vezes maior que a antiga versão, permite transferências simultâneas em ambos os sentidos, tornando-o um protocolo *full-duplex*. Uma outra solução será utilizar o barramento Ethernet 10GE ou 100GE uma vez que conseguem atingir velocidades bastante elevadas.

Apêndice A

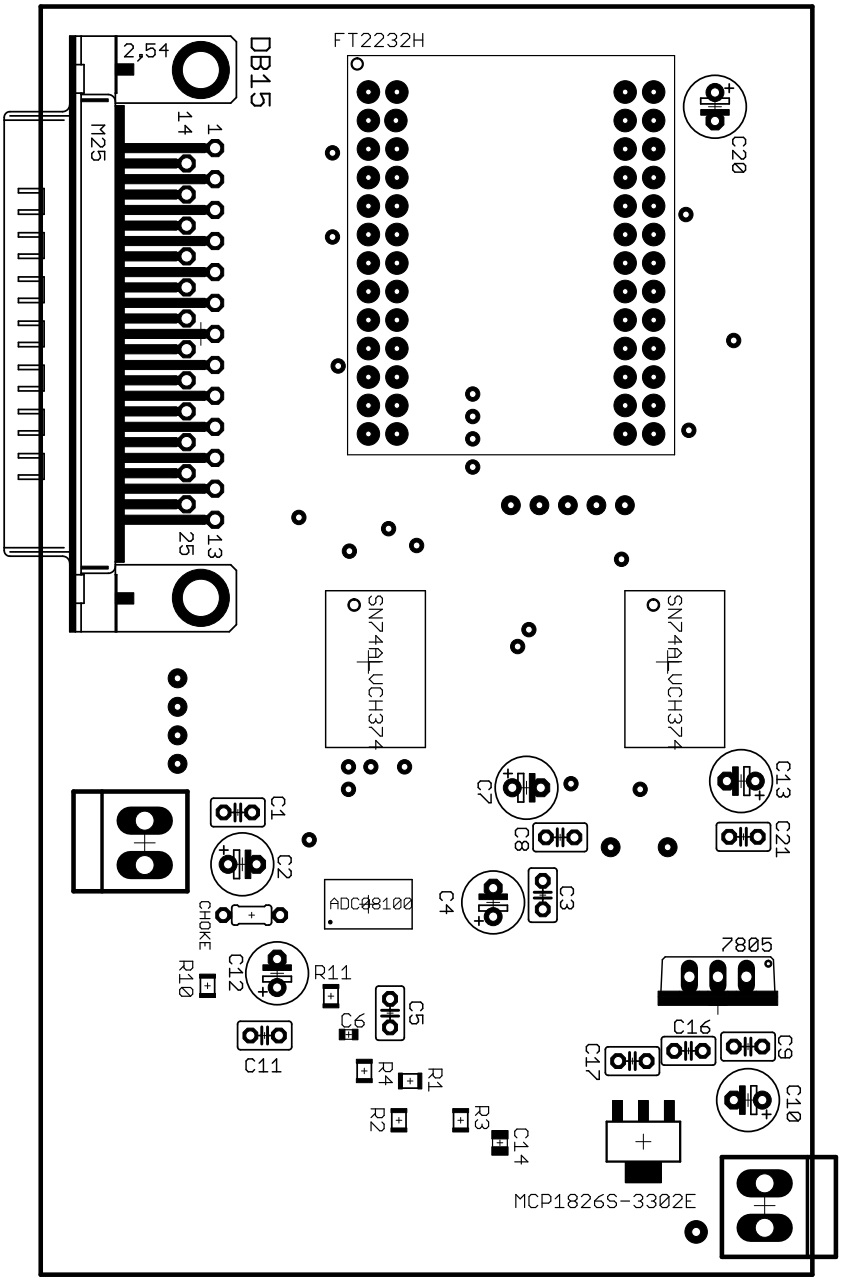
Esquemas eléctricos

A.1 Placa com a ADC08100

A.1.1 Esquema eléctrico da placa com a ADC08100

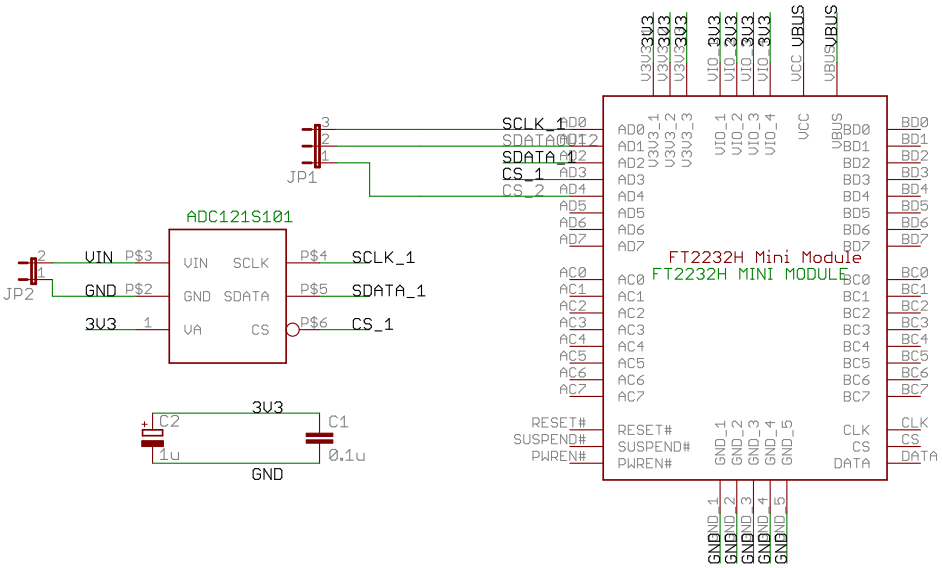

FT2232H Mini Module
ADC08100
7805
MCP1826S-3302E
DB15
J1
J4
+3V3
+5V
GND

A.1.2 Layout da PCB da placa com a ADC08100

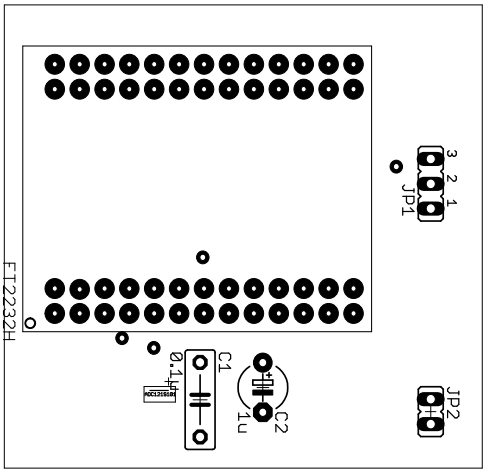


A.2 Placa com a ADC121S101

A.2.1 Esquema eléctrico da placa com a ADC121S101



A.2.2 Layout da PCB da placa com a ADC121S101



Apêndice B

Biblioteca NPLOT

A biblioteca NPLOT permite fazer plots de dados. É suportado pelo .NET Framework e portanto pode ser utilizado por qualquer língua desde que suportada pela .NET Framework, tal como é o caso da língua Visual C++/CLI.

Para utilizar esta biblioteca é necessário usar o namespace NPlot declarando-o da seguinte maneira:

```
using namespace NPlot
```

Depois de declarada pode ser usada para fazer plots 2D. Para um melhor entendimento do funcionamento desta biblioteca, será dado um exemplo pratico de como o utilizar esta biblioteca. Para tal será utilizado o programa Visual C++ 2008 Express Editon, no entanto poderá ser utilizado em outro programa de desenvolvimento que seja compatível com o .NET Framework.

Primeiro criou-se um projecto a partir do template do Windows Forms Application como se pode ver na figura B.1, e deu-se lhe o nome de NPLOT_EX.

Depois de criado o projecto será necessário incluir a biblioteca NPlot. Para tal é preciso adicionar a Toolbox um item, tal como se pode ver na figura B.2. Uma vez dentro deste menu é necessário ir a pasta que contem a ultima versão do NPLOT, e percorrer o seguinte caminho nplot-0.9.10.0\bin\net\2.0\release de modo a poder ser adicionado o DLL que esta nessa pasta.

Uma vez adicionado este DLL deverá aparecer na ToolBox o mesmo que o apresentado na figura B.3.

Para que seja incluído no desenho gráfico apenas é necessário arrastar o icon que está na Toolbox e ajusta-lo da maneira que se queira. No obstante apesar de não aparecer uma pre-visualização deste componente ele funcionará correctamente se adicionado o seguinte código ao ficheiro Form1.h.

```
using namespace NPlot
...
ArrayList^ XDAT;
ArrayList^ YDAT;
LinearAxis^ yAxis;
LinearAxis^ xAxis;
```

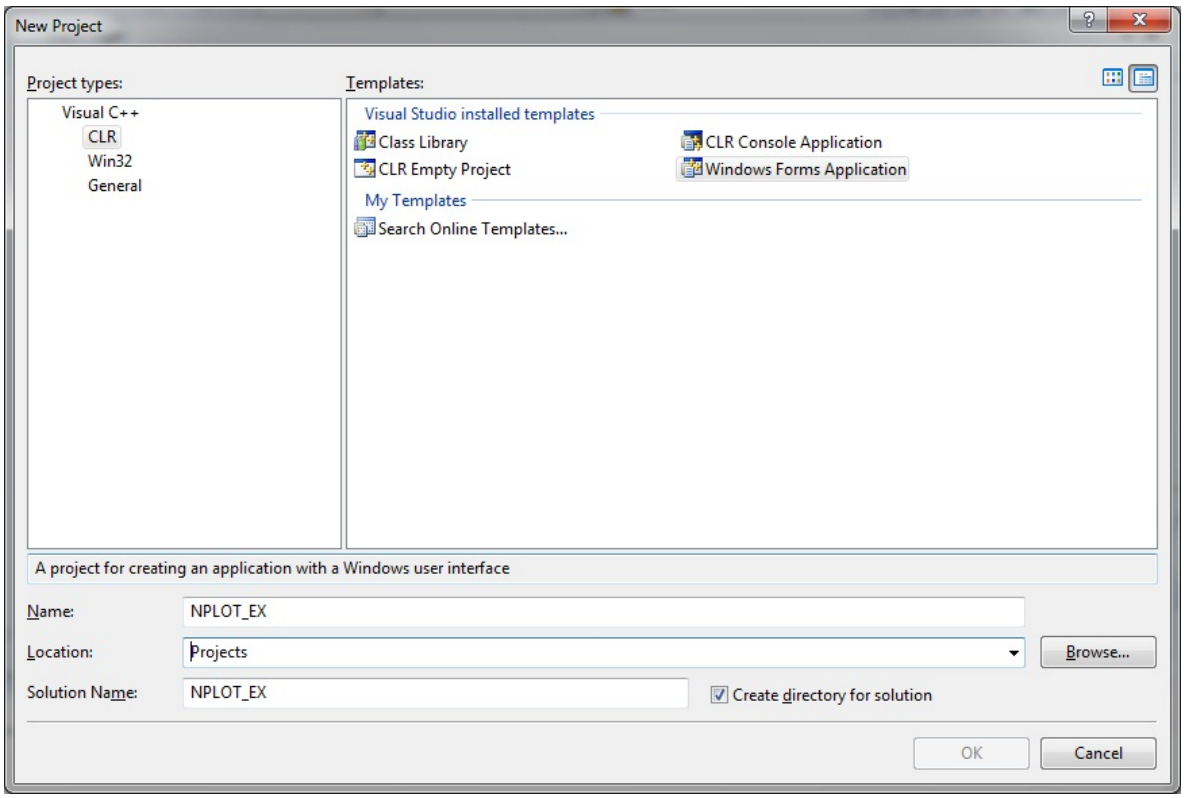


Figura B.1: Criar um novo projecto

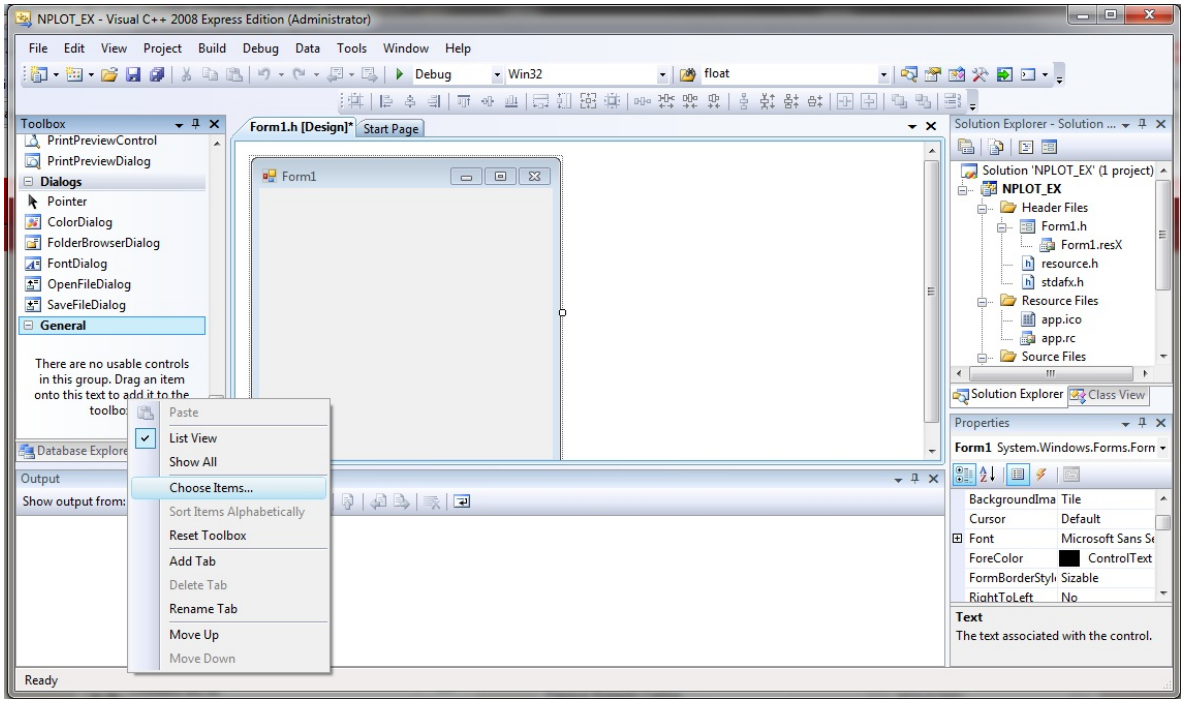


Figura B.2: Adicionar a biblioteca NPLOT

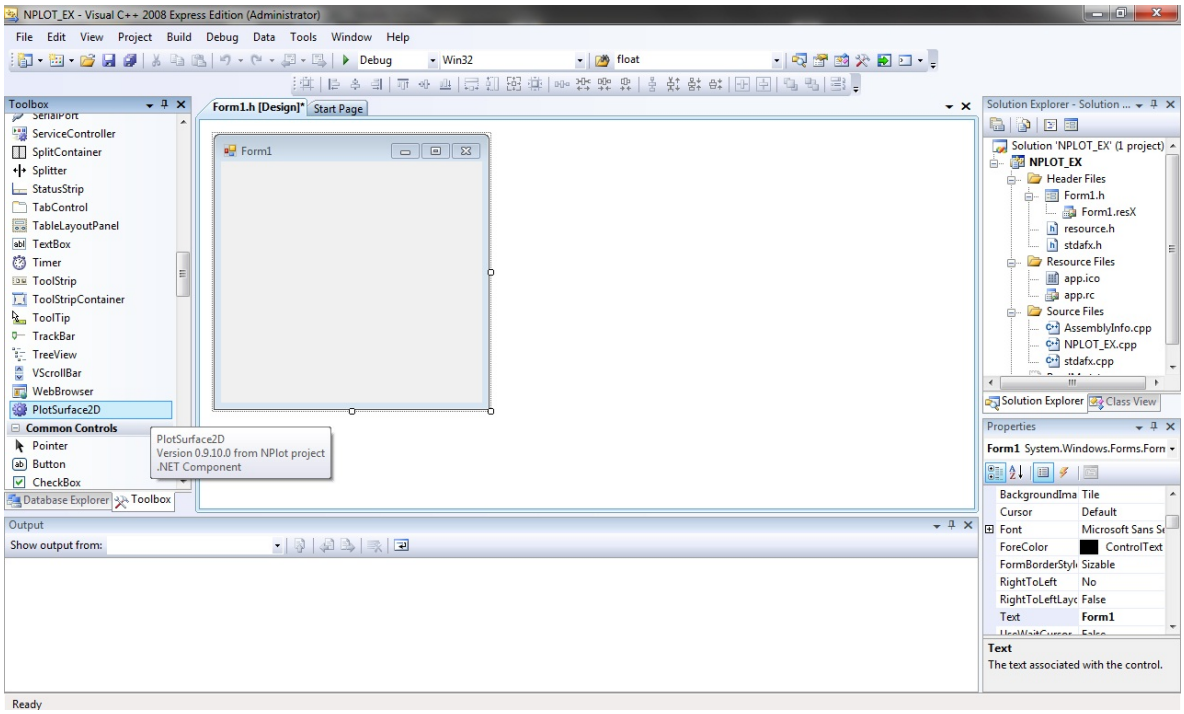


Figura B.3: Resultado da adição da biblioteca NPLOT

```
NPlot::LinePlot^ Linha;
Form1(void)
{
InitializeComponent();
//
//TODO: Add the constructor code here
//

XDAT = gcnew ArrayList;
YDAT = gcnew ArrayList;
Linha = gcnew NPlot::LinePlot;

plotSurface2D1->Clear(); // Apaga os dados do plot
plotSurface2D1->BackColor = Color::Empty; // A back color é colocada a Empty

for (int i = 0;i<=10-1;i++) //prenche os dados
{
XDAT->Add(i); //XDAT preenchidos com o valor de i
YDAT->Add(i); //YDAT preenchidos com o valor de i
}

//Inicializa a linha de dados
Linha->AbscissaData = XDAT; // Abscissas -> XDAT
Linha->OrdinateData = YDAT; // Ordenadas -> YPEAK
```

```
Linha->Color = Color::Blue; // Cor da linha -> Blue

plotSurface2D1->Add(Linha); // Adiciona a linha ao grafico

plotSurface2D1->Refresh(); // A janela do plot é actualizada
}
```

O resultado final é apresentado na figura B.4

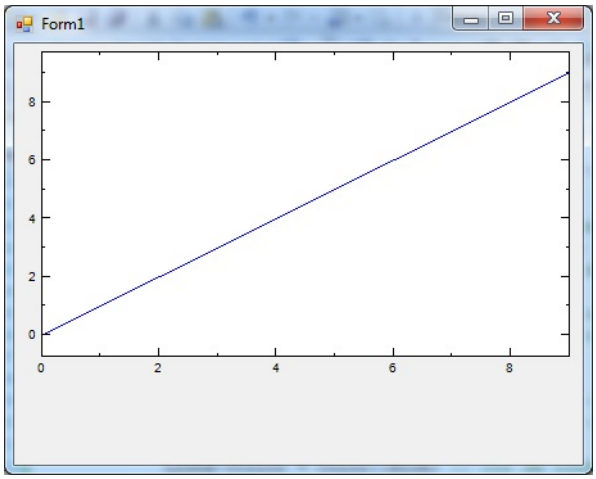


Figura B.4: Resultado final

Apêndice C

Manual de utilização do interface gráfico

Este anexo dedica-se ao manual de utilização do interface gráfico.

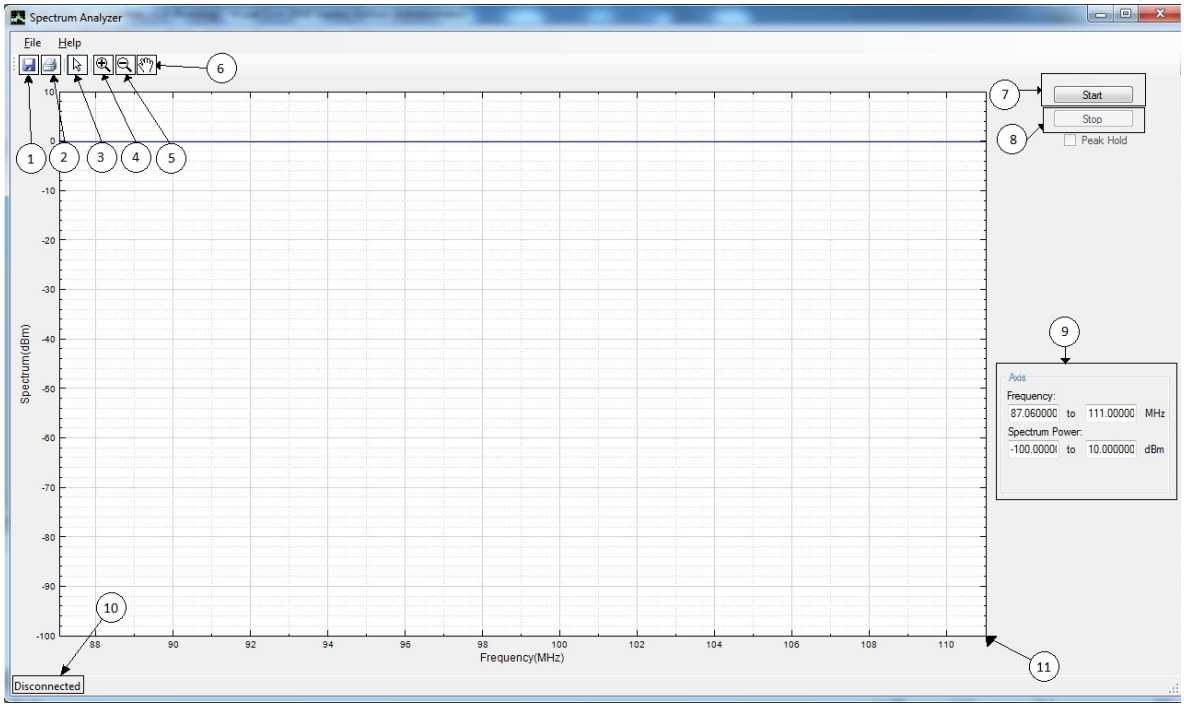


Figura C.1: Interface gráfico

Na figura C.1 pode-se ver o interface gráfico criado no âmbito desta dissertação. A figura tem a seguinte legenda:

1. Botão de guardar: Tem o propósito de guardar numa imagem o conteúdo actual da janela de plot que contêm a informação espectral;

2. Botão Imprimir: Imprime a informação espectral actual;

3. Pointer: Selecciona o pointer, não sendo possível efectuar qualquer acção sobre a janela de plot.
4. Zoom in: Permite ao utilizador fazer zoom in a janela de observação;
5. Zoom in: Permite ao utilizador fazer zoom out na janela de observação;
6. Hand: Premite ao utilizador deslocar a janela de observação;
7. Botao Start: Permite ao utilizador, começar a observar o espectro.
8. Botão Stop: Permite ao utilizador para a observação do espectro.
9. Axis: Premite ao utilizador seleccionar as frequências e gama de potência quer observar.
10. Estado: Transmite ao utilizador a informação sobre o estado de conexão com o front-end. Pode tomar o estado de disconnected e connected.
11. Janela de plot: Janela que contêm a informação actual do espectro.

Na figura reffig:igf pode-se ver o interface gráfico em funcionamento. O status do sistema como se pode ver é: Connected, significando portanto que o front-end está conectado ao PC.

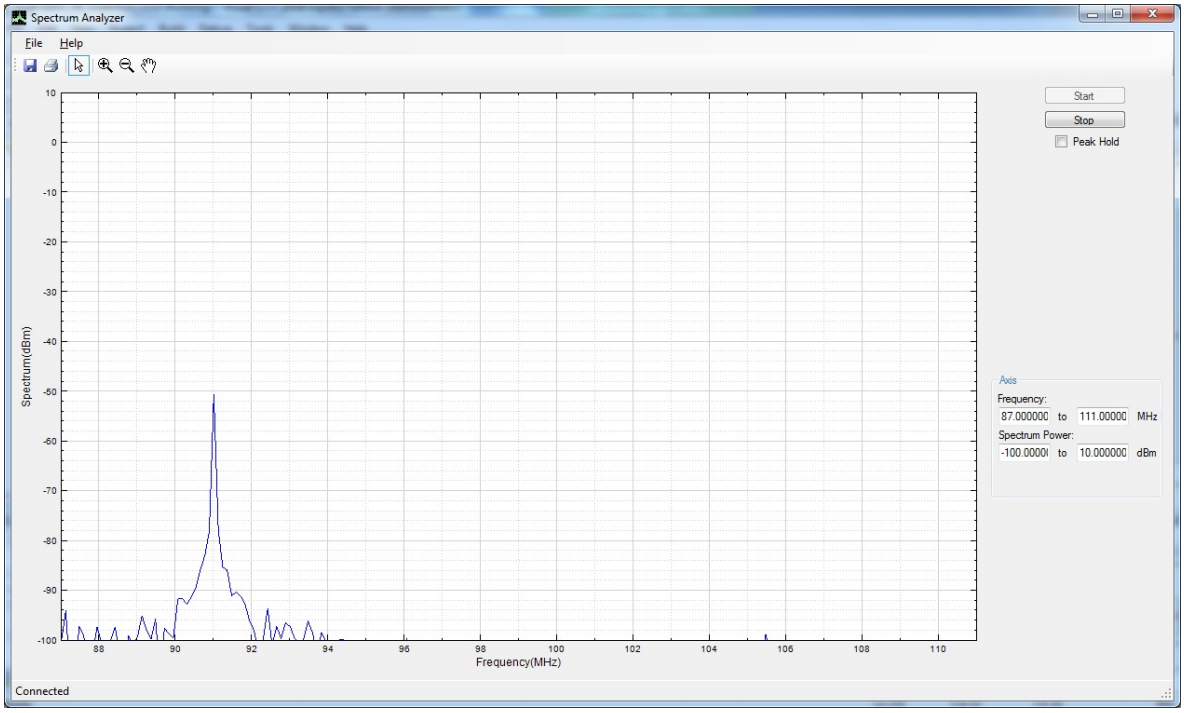


Figura C.2: Interface gráfico me funcionamento

Apêndice D

Sintetizador

Não é do âmbito desta dissertação a escolha do sintetizador utilizado na primeira parte desta dissertação. No entanto é necessário saber quais as suas características e como funciona.

O sintetizador escolhido foi o Si4136 que é capaz de sintetizar frequências entre 2025MHz e 2500MHz dividido em duas bandas. Tem integrados dois VCOs e todos os circuitos que permitem o seu correcto funcionamento. Na figura D.1 pode-se ver o diagrama funcional do chip onde estão incluídos os registos R e N para as duas bandas e ainda os VCOs e o PD. Para programar os registo o chip utiliza o interface SPI.

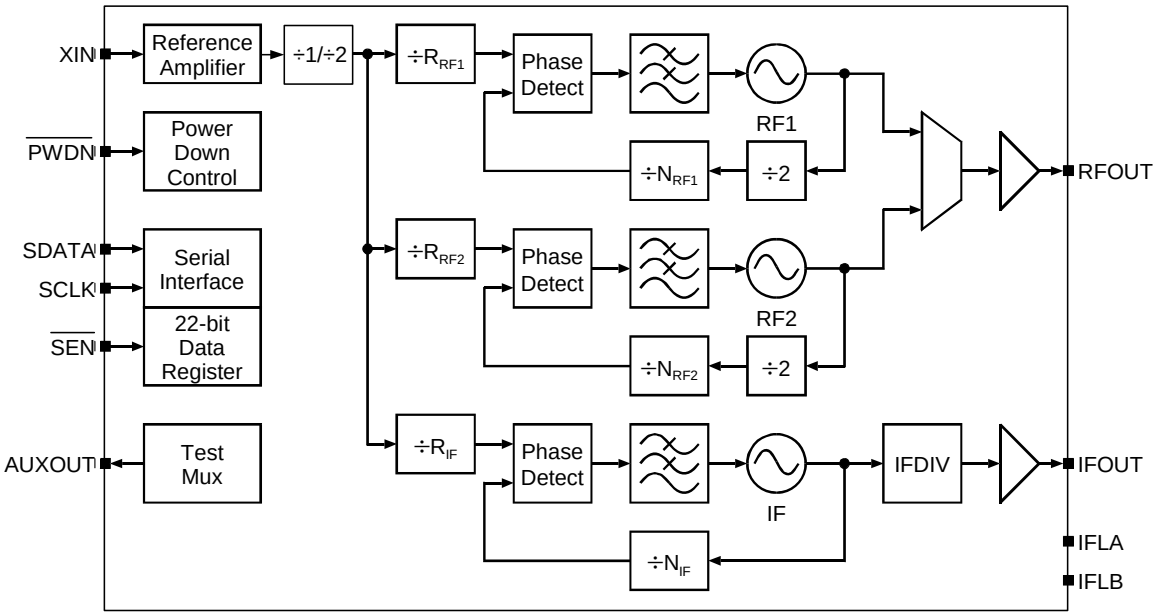


Figura D.1: Diagrama funcional do Si4136

Como o chip tem dois VCOs para se poder sintetizar uma frequência é necessário saber qual escolher. Cada VCO tem associado um registo N e R. A sub-banda inferior representada por RF1 vai desde 2300MHz a 2500MHz. As restrições dos seus registos são: o registo R toma os valores de 7 a 8189 e do N é a gama 992 a 65535. Para a sub-banda RF2 que vai desde 2025MHz a 2300MHz os registos têm as seguintes limitações: R pode variar de 7 a 8189 e o

registo N pode variar entre 240 e 32767.

Quando escrito em algum dos registos de uma sub-bandas, essa sub-banda é a que fica seleccionada. No entanto, para que seja sintetizada uma frequência é necessário que o lock do sintetizador seja atingindo. Tal acontece internamente sem que seja necessário configurar mais nenhum registo.

Bibliografia

[1] Quadro Nacional De Atribuição De Frequências. http://www.anacom.pt/streaming/qnaf06_versao_int.pdf?contentId=313152&field=ATTACHED_FILE.

[2] João Carreira. "Implementação em hardware de um analisador de espectros baseado em SDR". Master's thesis, Universidade Aveiro, 2010.

[3] J Mitola. "The Software Radio Architecture". *IEEE Communications Magazine*, 33(5):26–38, Maio 1995.

[4] D Albuquerque. "Front-Ends for Software-Defined Radio". *Department of Electronics, Telecommunications and Informatics*, 2009.

[5] Lúcia Margarida da Mata Antunes. "Software Defined Radio em FPGA". Master's thesis, Universidade Aveiro, 2009.

[6] Peter G. Cook and Wayne Bonser. "Architectural Overview of the SPEAKeasy System". *IEEE Journal on Selected Areas in Communications*, 17(4):650–661, Abril 1999.

[7] Pedro Miguel Duarte Cruz. "Caracterização de Sistemas para Software Defined Radio". Master's thesis, Universidade Aveiro, 2008.

[8] Universal Software Radio Peripheral - The Foundation for Complete Software Radio Systems. http://www.ettus.com/downloads/ettus_ds_usrp_v7.pdf.

[9] USRP2 - The Next Generation of Radio Systems. http://www.ettus.com/downloads/ettus_ds_usrp2_v5.pdf.

[10] FlexRadio Systems - Flex-5000. http://www.flex-radio.com/Products.aspx?topic=f5k_features.

[11] SDR-5001- Radio Transceiver Unit. http://www.spectrumsignal.com/products/pdf/sdr_5001.pdf.

[12] Buracchini, E. . "The software radio concept". *Communications Magazine, IEEE*, 38(9):138–143, Setembro 2000.

[13] Walter Tuttlebee. *"Software Defined Radio: Enabling Technologies"*. Wiley, 2002.

[14] Walt Kester. "Which ADC Architecture Is Right for Your Application". *Analoguen Dialogue 39-06*, Junho 2001.

[15] Waveform Measurement, Analysis Technical Committee of the IEEE Instrumentation, and Measurement Society. IEEE standard for terminology and test methods for analog-to-digital converters. *IEEE Std 1241-2000*, 2001.

[16] Waveform Measurement, Analysis Technical Committee of the IEEE Instrumentation, and Measurement Society. IEEE standard for digitizing waveform recorders. *IEEE Std 1057-1994*, 1994.

[17] N. Manicka. "GNU Radio Testbed". Master's thesis, Faculty of the University of Delaware, 2007.

[18] III Mitola, J. and Jr. Maguire, G.Q. Cognitive radio: making software radios more personal. *Personal Communications, IEEE*, 6(4):13 –18, August 1999.

[19] J. Guenin. "Jim Hoffmeyer on Overview of SCC41". *http://www.scc41.org/*, Setembro 2007.

[20] S. Lawrence Marple Jr. *"Digital Spectral Analysis - with applications"*. PRENTICE-HALL, INC., 1987.

[21] UM245R USB - Parallel FIFO Development Module Datasheet. `http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_UM245R.pdf`.

[22] FT2232H Mini Module - USB Hi-Speed FT2232H Evaluation Module Datasheet. `www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_FT2232H_Mini_Module.pdf`.

[23] FT2232H Dual High Speed USB to Multipurpose UART/FIFO IC. `http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT2232H.pdf`.

[24] Application Note AN-114 - FTDI Hi - Speed USB to SPI Example. `www.ftdichip.com/Support/Documents/AppNotes/AN_114_FTDI_Hi_Speed_USB_To_SPI_Example.pdf`.

[25] Application Note AN-135 - FTDI MPSSE Basics. `http://www.ftdichip.com/Support/Documents/AppNotes/AN_135_MPSSE_Basics.pdf`.

[26] Application Note AN-113 - FTDI Hi - Speed USB to I2C Example. `www.ftdichip.com/Support/Documents/AppNotes/AN_111_FTDI_Hi_Speed_USB_To_I2C_Example.pdf`.

[27] Application Note AN-130 - FT2232H used in An FT245 Style Synchronous FIFO Mode. `www.ftdichip.com/Support/Documents/AppNotes/AN_130_FT2232H_Used_In_FT245SynchronousFIFOMode.pdf`.

[28] ADC08100 Datasheet. `http://www.national.com/ds/DC/ADC08100.pdf`.

[29] MAX3542 Datasheet. `http://datasheets.maxim-ic.com/en/ds/MAX3542.pdf`.

[30] SAW Components - Data Sheet X6966M. `http://www.epcos.com/inf/40/ds/mm/X6966M.pdf`.

[31] Software Application Development D2XX Programmer Guide. `www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer%27s_Guide%28FT_000071%29.pdf`.

[32] NPLOT. http://netcontrols.org/nplot/wiki/.

[33] ADC121S101 Datasheet. http://www.national.com/ds/DC/ADC121S101.pdf.

[34] KNJN LLC. http://www.knjn.com/FPGA-FX2.html.